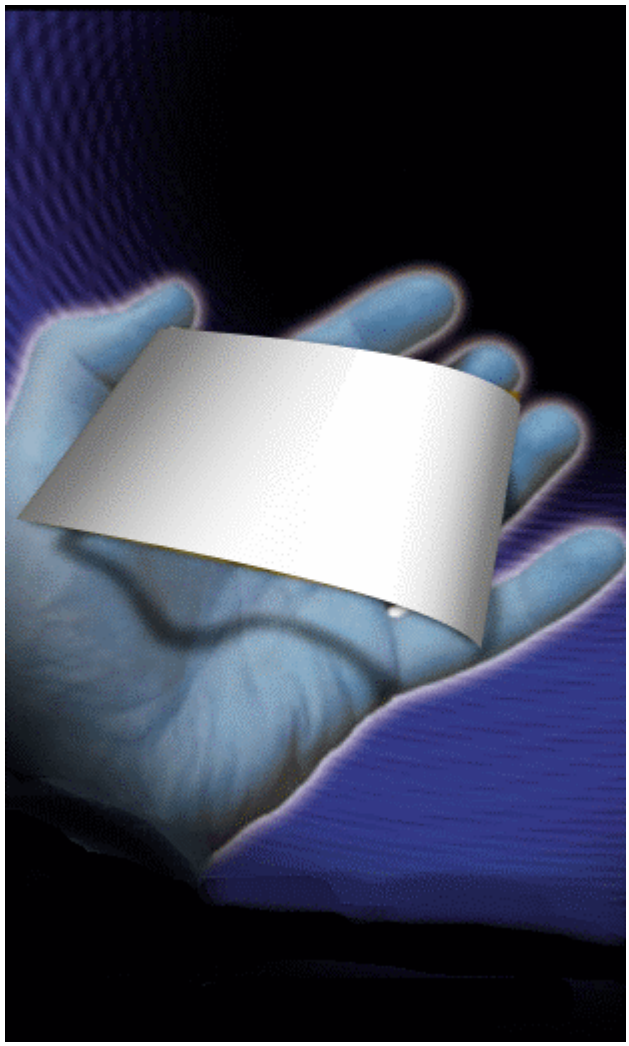
Texto que introduce al lector en el diseño y generación de informes con Crystal Reports para Visual Studio .NET, utilizando como base el lenguaje Visual Basic .NET.

Se cubren tanto aspectos elementales como avanzados, de manera que el lector pueda confeccionar variados tipos de informes en función de las diversas necesidades de impresión que precisan las aplicaciones.

Como requisitos previos, se recomienda el conocimiento del lenguaje Visual Basic .NET, así como estar familiarizado con el sistema operativo Windows a nivel de usuario.



CRYSTAL REPORTS PARA VISUAL STUDIO .NET (VISUAL BASIC .NET)

LUIS MIGUEL BLANCO

alhambra eidos

ADVERTENCIA LEGAL

Todos los derechos de esta obra están reservados a Grupo EIDOS Consultoría y Documentación Informática, S.L.

El editor prohíbe cualquier tipo de fijación, reproducción, transformación, distribución, ya sea mediante venta y/o alquiler y/o préstamo y/o cualquier otra forma de cesión de uso, y/o comunicación pública de la misma, total o parcialmente, por cualquier sistema o en cualquier soporte, ya sea por fotocopia, medio mecánico o electrónico, incluido el tratamiento informático de la misma, en cualquier lugar del universo.

El almacenamiento o archivo de esta obra en un ordenador diferente al inicial está expresamente prohibido, así como cualquier otra forma de descarga (downloading), transmisión o puesta a disposición (aún en sistema streaming).

La vulneración de cualesquiera de estos derechos podrá ser considerada como una actividad penal tipificada en los artículos 270 y siguientes del Código Penal.

La protección de esta obra se extiende al universo, de acuerdo con las leyes y convenios internacionales.

Esta obra está destinada exclusivamente para el uso particular del usuario, quedando expresamente prohibido su uso profesional en empresas, centros docentes o cualquier otro, incluyendo a sus empleados de cualquier tipo, colaboradores y/o alumnos.

Si Vd. desea autorización para el uso profesional, puede obtenerla enviando un e-mail guido.peterssen@alhambra-eidos.com o al fax (34) 91 7872301.

Si piensa o tiene alguna duda sobre la legalidad de la autorización de la obra, o que la misma ha llegado hasta Vd. vulnerando lo anterior, le agradeceremos que nos lo comunique al e-mail guido.peterssen@alhambra-eidos.com o al fax (34) 91 7872301. Esta comunicación será absolutamente confidencial.

Colabore contra el fraude. Si usted piensa que esta obra le ha sido de utilidad, pero no se han abonado los derechos correspondientes, no podremos hacer más obras como ésta.

© Luis Miguel Blanco, 2003
© Grupo EIDOS Consultaría y Documentación Informática, S.L., 2002

ISBN 84-88457-65-0

Crystal Reports para Visual Studio .NET (Visual Basic .NET)

Luis Miguel Blanco

Responsable de la edición
Guido Peterssen (guido.peterssen@alhambra-eidos.com)

Coordinación de la edición
Antonio Quirós
(antonio.quiros@alhambra-eidos.com)

Autoedición
Magdalena Marín (magdalena.marin@alhambra-eidos.com)
Luis Miguel Blanco (lmiguel.blanco@alhambra-eidos.com)

Grupo EIDOS
C/ Albasanz 16 4ª Planta
28037-Madrid (España)
Tel: (34) 91 7872300 Fax: (34) 91 7872301
www.grupoeidos.com/www.eidos.es
www.LaLibreriaDigital.com

Índice

ÍNDICE 5

ELABORACIÓN DE INFORMES CON CRYSTAL REPORTS PARA VISUAL STUDIO .NET (VISUAL BASIC .NET) 7

EL TRABAJO DE IMPRESIÓN DESDE WINDOWS 7
CRYSTAL REPORTS PARA VISUAL STUDIO .NET 7
ESPACIOS DE NOMBRES DE CRYSTAL REPORTS PARA VISUAL STUDIO .NET 8
CREACIÓN DE UN INFORME CON EL ASISTENTE DE CRYSTAL REPORTS 9
EL DISEÑADOR DE INFORMES 15
EL CONTROL CRYSTALREPORTVIEWER 16
LA CLASE DEL INFORME 18
TIPOS DE INFORME 20
Informe con establecimiento inflexible de tipos (Strong typed) 20
Informe sin tipo (Un-Typed) 20
Informe sin tipo cargado desde el control CrystalReportViewer 20
Informe sin tipo cargado desde el componente ReportDocument 21
CREACIÓN DE UN INFORME MANUAL A PARTIR DEL DISEÑADOR VACÍO 23
Seleccionar y establecer la conexión con un origen de datos 24
Agregar campos de la tabla al informe 25
Manipular objetos en el diseñador del informe 25
Aplicar formato a los campos 26
Completar el informe con controles adicionales 29
Fecha y hora del informe 30
Indicadores de página 31

Imagen 31
INFORME CON DATOS ORDENADOS 32
SELECCIONAR LOS DATOS A MOSTRAR EN EL INFORME 34
CREACIÓN DE GRUPOS DE REGISTROS EN EL INFORME 36
Incluir manualmente el grupo en el informe 39
INFORME CON TABLAS VINCULADAS 39
Tablas vinculadas con el asistente 40
Tablas vinculadas manualmente 43
Obtención de datos relacionados entre tablas vinculadas 45
FÓRMULAS 47
Suprimir valores duplicados 49
MODIFICACIÓN DE LA FÓRMULA DE SELECCIÓN 51
APLICAR FORMATO A UNA SECCIÓN 52
CREACIÓN DE TOTALES Y SUBTOTALES 54
Uso del asistente de informes para crear totales y subtotales 54
Creación manual de un total general 56
Creación manual de un subtotal 57
INCLUSIÓN DE UN GRÁFICO ESTADÍSTICO EN EL INFORME 60
MANIPULACIÓN POR CÓDIGO DE LOS ELEMENTOS DEL INFORME 63
TÍTULO DEL INFORME 63
SELECCIÓN DE REGISTROS POR CÓDIGO 65
ORDENACIÓN DE REGISTROS POR CÓDIGO 67
ESTABLECER POR CÓDIGO EL GRUPO DE REGISTROS 69
ENVIAR LOS DATOS DE CONEXIÓN AL INFORME POR CÓDIGO 71
MODIFICAR CAMPOS DE FÓRMULA Y TEXTO POR CÓDIGO 73
IMPRESIÓN DEL INFORME DESDE CÓDIGO 76
UTILIZAR UN PROCEDIMIENTO ALMACENADO PARA OBTENER LOS REGISTROS 77
PASO DE PARÁMETROS A UN INFORME 79
EMPLEAR UN DATASET EN EL INFORME PARA OBTENER LOS DATOS 81
CREAR UN INFORME A PARTIR DE UN ARCHIVO XML Y SU ESQUEMA CORRESPONDIENTE 85
EXPORTAR UN INFORME A OTROS FORMATOS DE ARCHIVO 89
MANIPULACIÓN POR CÓDIGO DEL CONTROL CRYSTALREPORTVIEWER 93
CREACIÓN DE INFORMES EN ASP.NET 96
INFORMES GESTIONADOS EN MEMORIA CACHÉ 98
ACTUALIZACIÓN DE LA VERSIÓN DEL INFORME MANTENIDO EN CACHÉ 101
PUBLICACIÓN DE UN INFORME COMO UN WEB SERVICE 105

1

Elaboración de informes con Crystal Reports para Visual Studio .NET (Visual Basic .NET)

El trabajo de impresión desde Windows

La generación de documentación impresa en forma de listados o informes es una de las características que se exigen a toda aplicación profesional.

Realizar este trabajo desde Windows, si no utilizamos la herramienta de programación adecuada, se convierte en una tarea ardua y complicada, que obliga al programador a emplear el API de Windows, teniendo que investigar con las múltiples funciones de impresión existentes, tipos de letra, tamaños de papel, etc.; haciendo pasar al sufrido desarrollador por un auténtico calvario, para finalmente, obtener un resultado en muchas ocasiones poco vistoso, comparado con el esfuerzo que ha sido necesario emplear para su consecución.

Crystal Reports para Visual Studio .NET

Crystal Reports (o CR, como también lo denominaremos a lo largo del texto) es una herramienta especializada en la generación de informes para el entorno Windows, que mediante un conjunto de diseñadores y asistentes nos permite la creación de los listados para nuestras aplicaciones de un modo muy sencillo, y con un resultado y apariencia muy profesionales.

Se trata de un generador de informes que ha acompañado a Visual Basic desde hace varias versiones, pero uno de sus principales inconvenientes (concretamente hasta Visual Studio 6, inclusive) radicaba en el hecho de que debía utilizarse como producto separado del IDE de VB, puesto que a pesar de incluirse en los CD's de instalación de Visual Studio, no se instalaba por defecto, lo que obligaba al programador a instalarlo y ejecutarlo de forma separada al IDE habitual de trabajo.

Esta falta de integración de CR con el IDE de desarrollo de Microsoft se ha solucionado con la llegada de la tecnología .NET, ya que Crystal Decisions, el fabricante de CR ha desarrollado una nueva versión de su generador de informes: Crystal Reports para Visual Studio .NET, que como su nombre indica, se encuentra totalmente integrada con el entorno de desarrollo de VS.NET; prueba de ello reside en el hecho de que CR se instala ahora junto con el resto de elementos del IDE de Visual Studio, durante el proceso de instalación del entorno de desarrollo de .NET Framework.

La ventaja más inmediata pues que obtenemos de esta versión de CR, reside en que al funcionar de forma integrada con VS.NET, el motor de informes del generador se coordina de manera más eficiente con el resto de clases de la plataforma. Todas las características de CR están disponibles ahora a través de una serie de clases, organizadas en un conjunto de espacios de nombres, de los cuales, el principal es CrystalDecisions. Esta integración da como fruto una mejora en aquellos aspectos ya conocidos de versiones anteriores de CR, facilitando el uso de informes en aplicaciones web a través de su visualización en Web Forms, destacando también como novedad, la posibilidad de publicar nuestros informes como Web Services.

La compatibilidad hacia atrás también está contemplada, soportando los informes creados con versiones anteriores de CR, e incluso con el DataReport de VB6.

A lo largo de los siguientes apartados, describiremos los principales puntos a tener en cuenta para la elaboración de informes sencillos, abordando también algunos aspectos avanzados, que proporcionen al lector el punto de partida para la creación de los listados para sus aplicaciones.

Espacios de nombres de Crystal Reports para Visual Studio .NET

Crystal Reports ha sido remodelado completamente para esta versión de .NET, adaptándolo a la arquitectura del Framework de Microsoft. Aparte de las mejoras introducidas en elementos visuales como asistentes y diseñadores de este generador de informes, se ha mejorado notablemente el apartado de manipulación del informe a través del código de la aplicación. Por todos estos motivos, la actual versión de CR está compuesta por un conjunto de tipos que se organizan en los siguientes espacios de nombres.

- **CrystalDecisions.CrystalReports.Engine**. Contiene las clases que representan al motor de informes del generador, y nos permiten el manejo de la mayoría de sus aspectos por código.

- **CrystalDecisions.ReportSource**. Contiene los tipos que actúan como capa de negocio entre los visualizadores de informes para las interfaces Windows o Web y el motor de CR, procesando las peticiones de estos visualizadores para obtener los datos a través del motor y mostrarlos.

- **CrystalDecisions.Shared**. Alberga un conjunto de tipos que son utilizados de forma compartida por las clases del motor de informes y los visualizadores de CR.

- **CrystalDecisions.Web**. Conjunto de clases relacionadas con el visualizador de informes CrystalReportViewer, en su versión para aplicaciones ASP.NET:

- **CrystalDecisions.Web.Services, CrystalDecisions.Web.Services.Enterprise**. Estos espacios de nombres contienen las clases que permiten la publicación de un informe como un Web Service.

- **CrystalDecisions.Windows.Forms**. Conjunto de clases relacionadas con el visualizador de informes CrystalReportViewer, en su versión para aplicaciones Windows.

Creación de un informe con el asistente de Crystal Reports

En este apartado vamos a describir el proceso de elaboración de un informe sencillo utilizando el asistente de generación de informes que proporciona CR. De esta manera demostraremos cuán fácil y rápida es la creación de un listado básico que no requiera de excesivos ornamentos, pero que suministrará la suficiente información al usuario.

En primer lugar, iniciaremos Visual Studio .NET en el caso de que no lo tengamos ya en ejecución, y crearemos un nuevo proyecto de tipo Aplicación Windows con el nombre RepSencillo (para acceder al proyecto de este ejemplo hacer clic aquí)

Todo informe debe ser *alimentado* con información desde una fuente de datos, por lo que tanto para el ejemplo de este apartado como para el resto de ejemplos expuestos utilizaremos SQL Server como servidor de datos, y su base de datos Northwind, que se incluye de muestra para realizar pruebas cuando instalamos este gestor de bases de datos. Igualmente, todos los proyectos de ejemplo que vayamos desarrollando serán de tipo Aplicación Windows.

A continuación añadiremos un informe al proyecto mediante el menú de VS.NET *Proyecto + Agregar nuevo elemento*, seleccionando en el cuadro de diálogo del mismo nombre el icono Crystal Report, y asignando el nombre rptPrueba al nuevo informe. Ver la Figura 1.

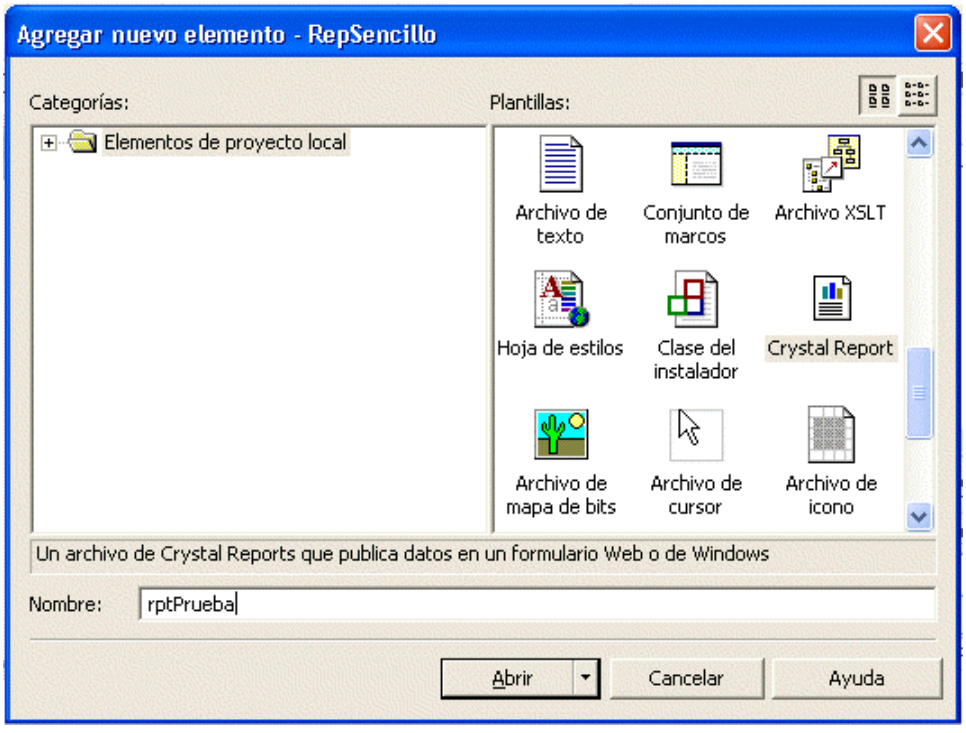


Figura 1. Crear nuevo informe en el proyecto.

Esta acción creará un nuevo informe que será guardado en un archivo con el mismo nombre que hemos dado al informe, y la extensión .RPT.

Es posible que en este momento aparezca un asistente para registrarnos a través de Internet, como usuarios de Crystal Reports para Visual Studio .NET. Ver la Figura 2.

Figura 2. Asistente para el registro de Crystal Reports.

Esta operación la podemos realizar en este momento o posponerla, aunque es recomendable efectuar el registro, ya que es gratuito y una vez introducidos los códigos que nos suministra el fabricante, evitaremos que se muestre esta ventana de recordatorio cada vez que vamos a utilizar un informe del proyecto. Ver la Figura 3.

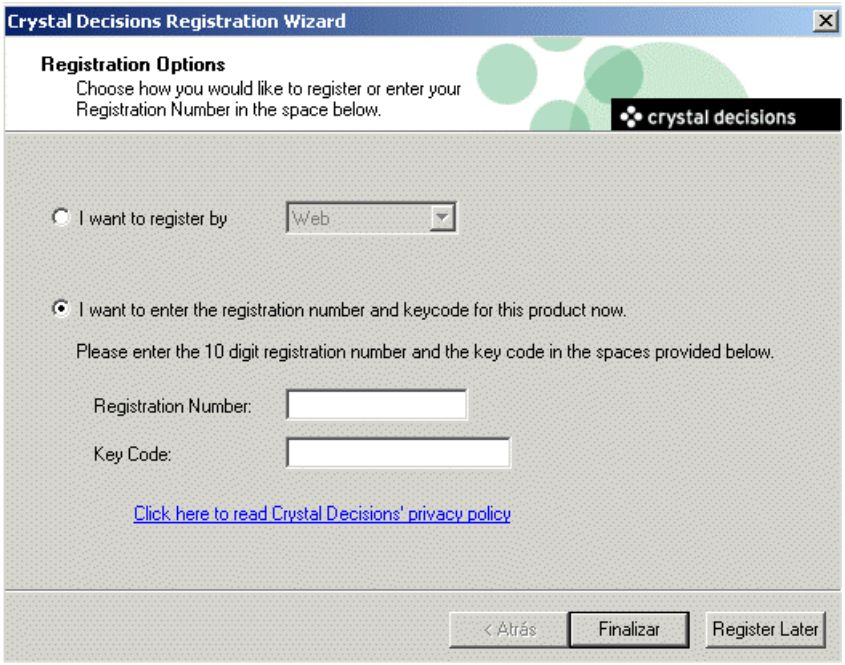


Figura 3. Introducción de códigos de registro de Crystal Reports.

Independientemente de si registramos o no Crystal Reports, se mostrará a continuación la ventana *Galería de Crystal Reports*, en la que seleccionaremos cómo queremos crear el informe, y la modalidad de asistente que vamos a utilizar, en el caso de que vayamos a emplear uno. Dejaremos los valores por defecto que se proporcionan en esta ventana: para crear un nuevo informe usaremos el asistente, y como tipo de asistente emplearemos el Estándar, que vemos en la Figura 4.

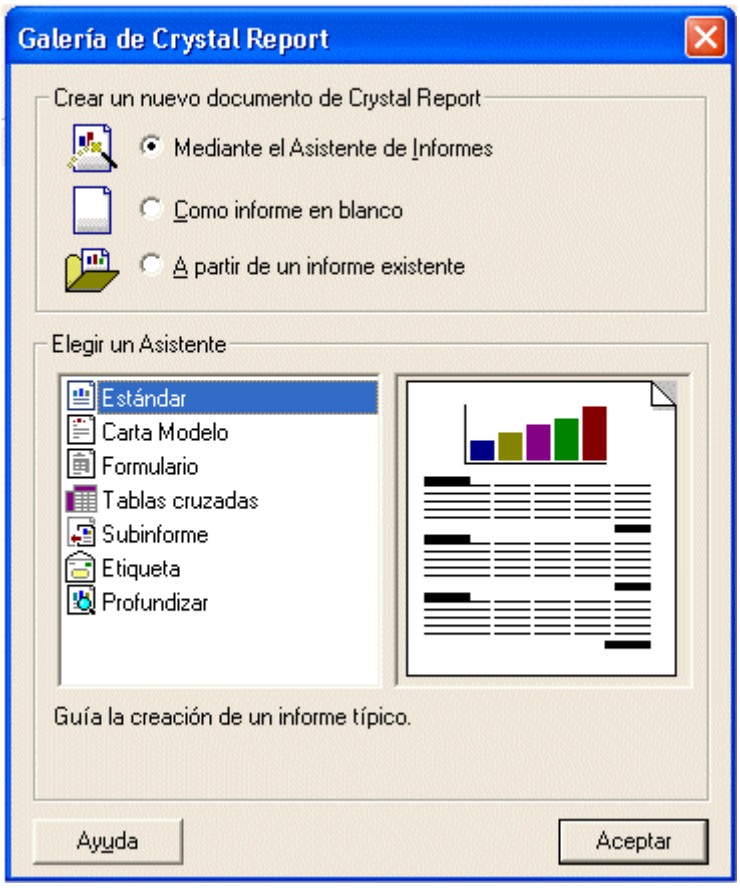


Figura 4. Selección del modo de creación y asistente del informe.

Tras pulsar el botón Aceptar de esta ventana se iniciará el asistente de Crystal Reports para la creación del informe. Para el presente ejemplo nos ocuparemos sólo de los pasos principales de dicho asistente, dejando el resto para otros informes que requieran de diferentes niveles de complejidad.

En el primer paso, correspondiente a la pestaña Datos, seleccionaremos el origen de datos que va a utilizar el informe. Ver la Figura 5.

Debemos hacer clic en el nodo OLE DB (ADO) del panel izquierdo de la ventana, lo que nos llevará a su vez a otro asistente para configurar la conexión a datos, en el que seleccionaremos el proveedor de OLE DB, en nuestro caso el correspondiente a SQL Server, como muestra la Figura 6.

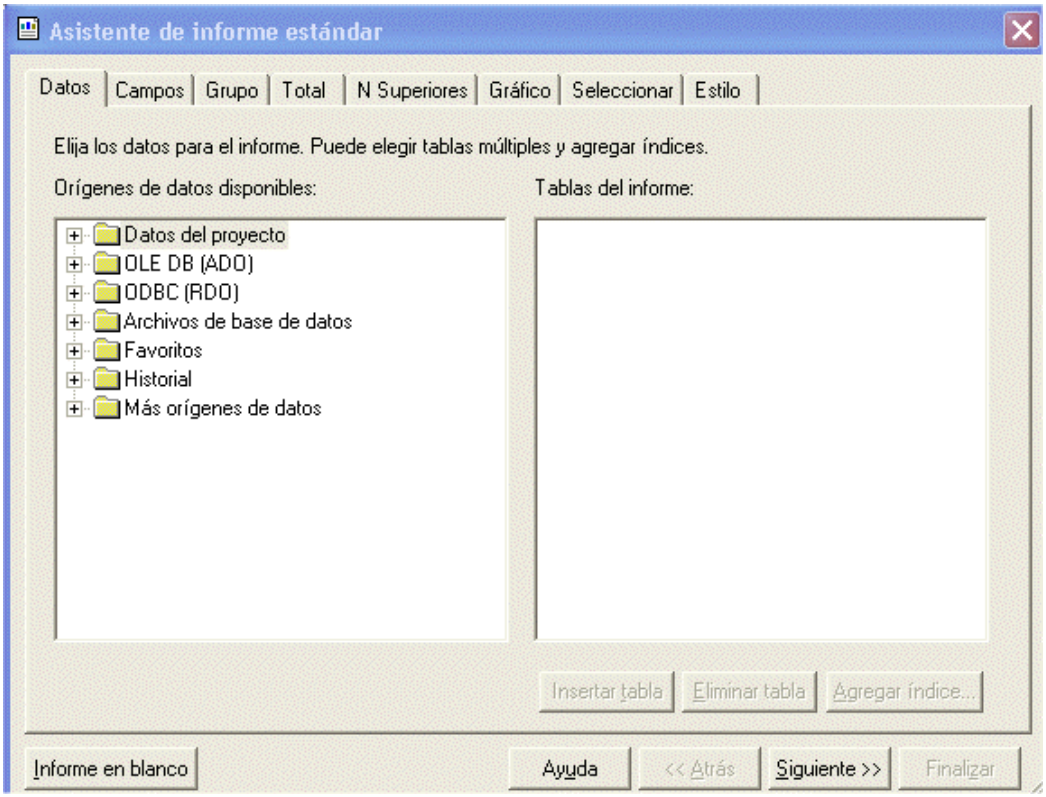


Figura 5. Selección del origen de datos para el informe.

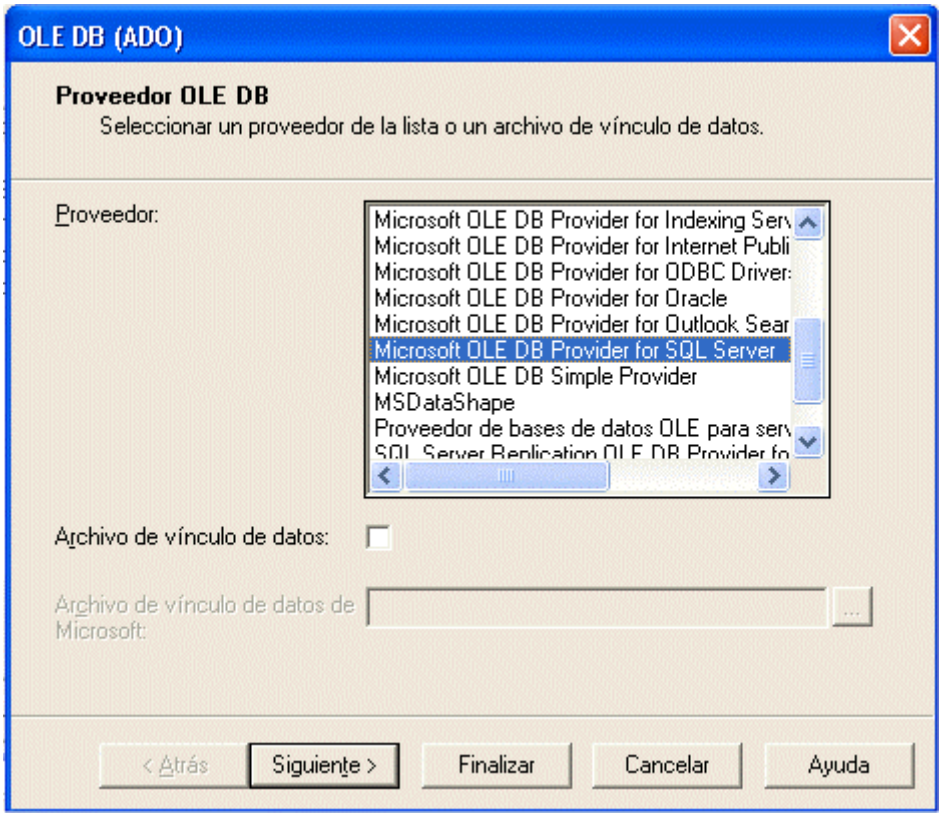


Figura 6. Selección del proveedor de OLE DB para el informe.

Pulsando el botón Siguiente pasaremos a introducir los valores de conexión con el servidor de base de datos, que serán los siguientes:

- **Servidor**. localhost.
- **Id. del usuario**. sa.
- **Contraseña**. No escribir nada en este campo.
- **Base de datos**. Northwind.

La Figura 7 muestra esta ventana.

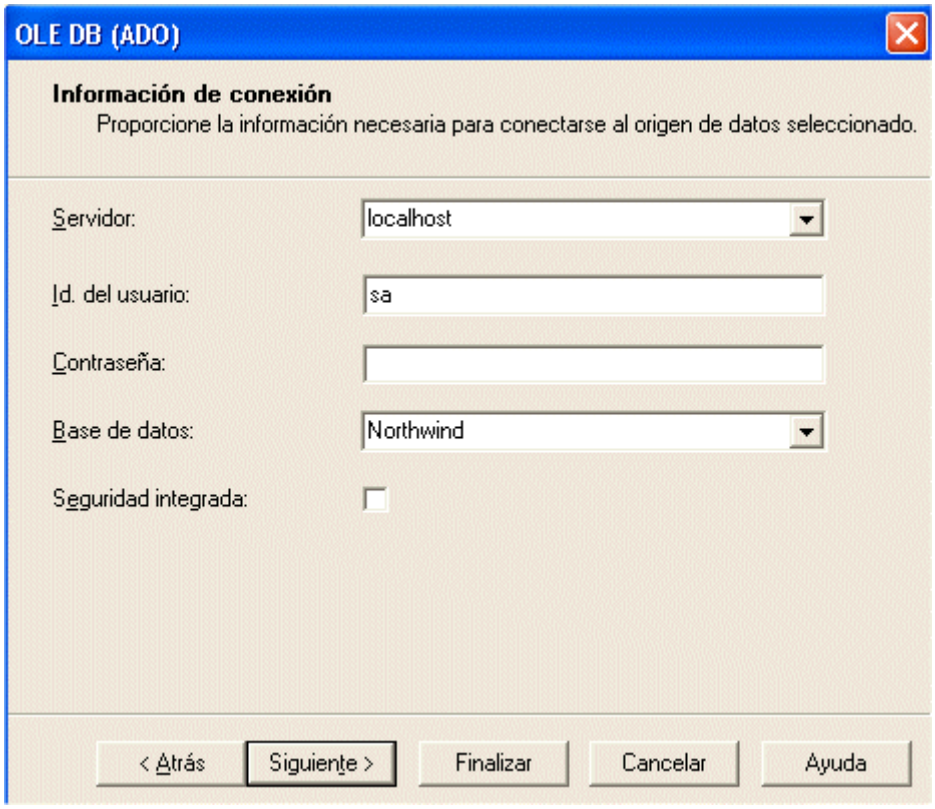


Figura 7. Valores de conexión para el informe.

Volviendo a pulsar Siguiente, visualizaremos información adicional sobre la conexión. Ver la Figura 8.

Finalizaremos este asistente de conexión a la base de datos con lo que volveremos al asistente principal, en el que iremos expandiendo la conexión localhost, recién creada, pasando por la base de datos Northwind, hasta llegar a la tabla Employees, que será la que utilicemos para el informe. Pulsando el botón Insertar tabla, añadiremos dicha tabla al informe. Ver la Figura 9.

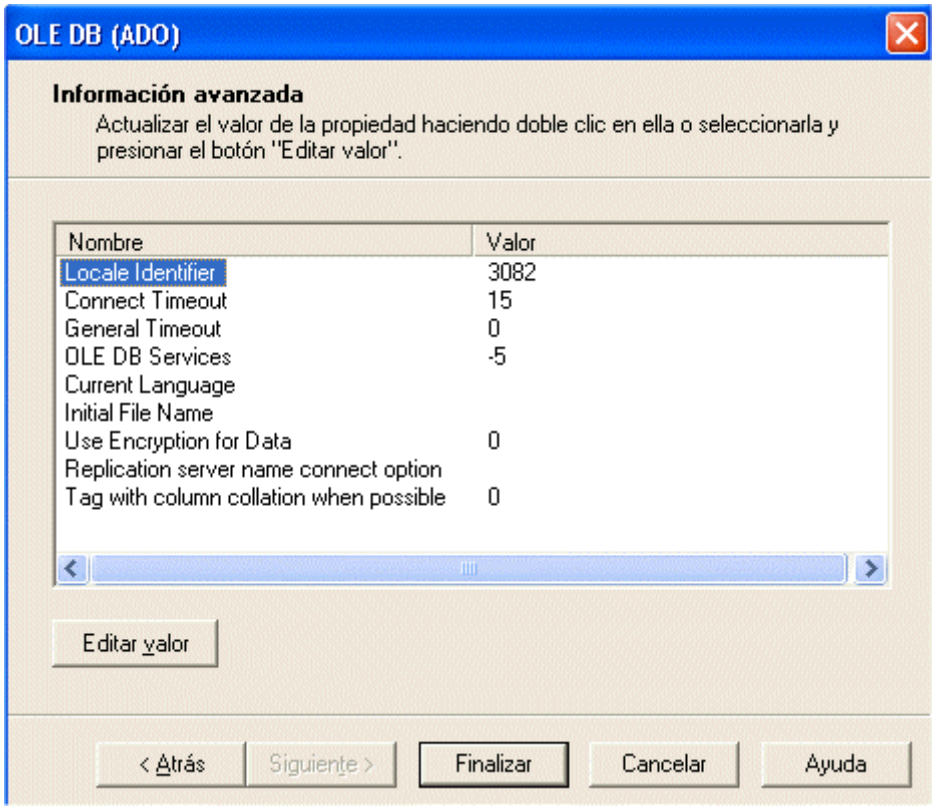


Figura 8. Información avanzada sobre la conexión.

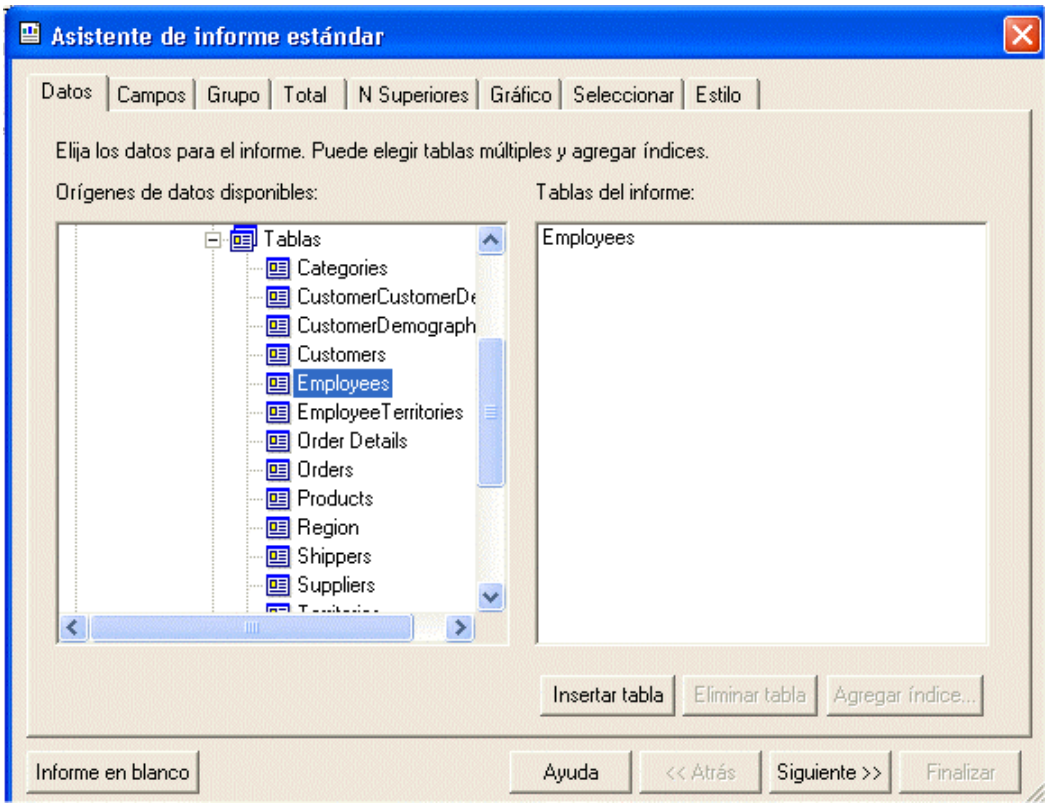


Figura 9. Selección de tabla de la base de datos, para el informe.

En el siguiente paso seleccionaremos los campos que va a mostrar el listado. Podemos seleccionar uno a uno, o en este caso, y para simplificar, pulsaremos el botón *Agregar todos*, que añadirá la totalidad de campos de la tabla. Posteriormente quitaremos algunos, seleccionándolos del panel de campos añadidos y pulsando el botón Quitar. Ver la Figura 10.

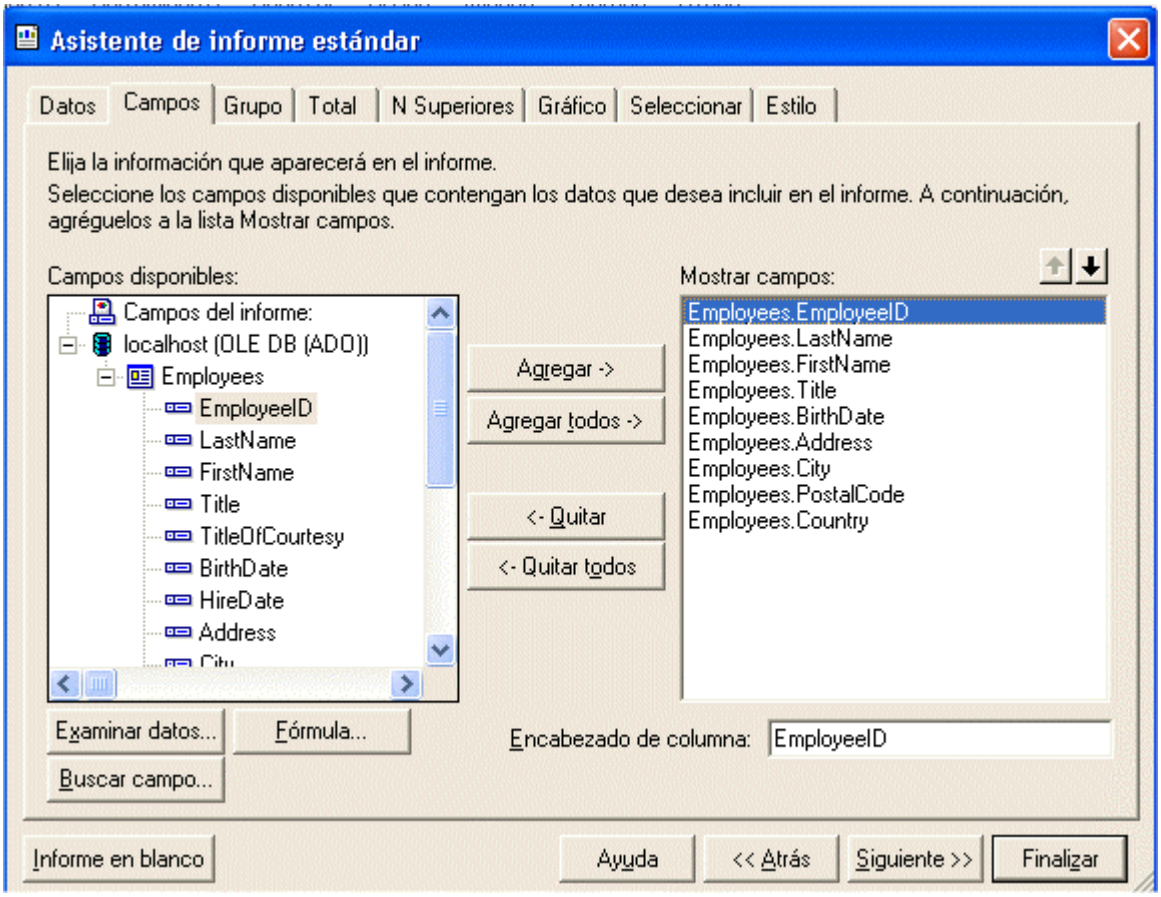


Figura 10. Insertar campos para el informe.

Con estos pasos habríamos terminado de crear nuestro informe básico, por lo que ahora, para finalizar el asistente, pulsaremos el botón Finalizar de su ventana, pasando al diseñador visual del informe.

El diseñador de informes

Completados los pasos del asistente para crear el informe explicados en el apartado anterior, obtendremos el nuevo informe en la ventana correspondiente al diseñador del informe, que podemos ver en la Figura 11.

Esta ventana nos permite el diseño visual del informe de una manera muy fácil y cómoda, con el mismo estilo de trabajo que el usado con el diseñador de formularios Windows.

Podemos comprobar que el asistente ha creado un conjunto de controles dentro del diseñador en forma de campos, que mostrarán los datos cuando sea ejecutado el informe.

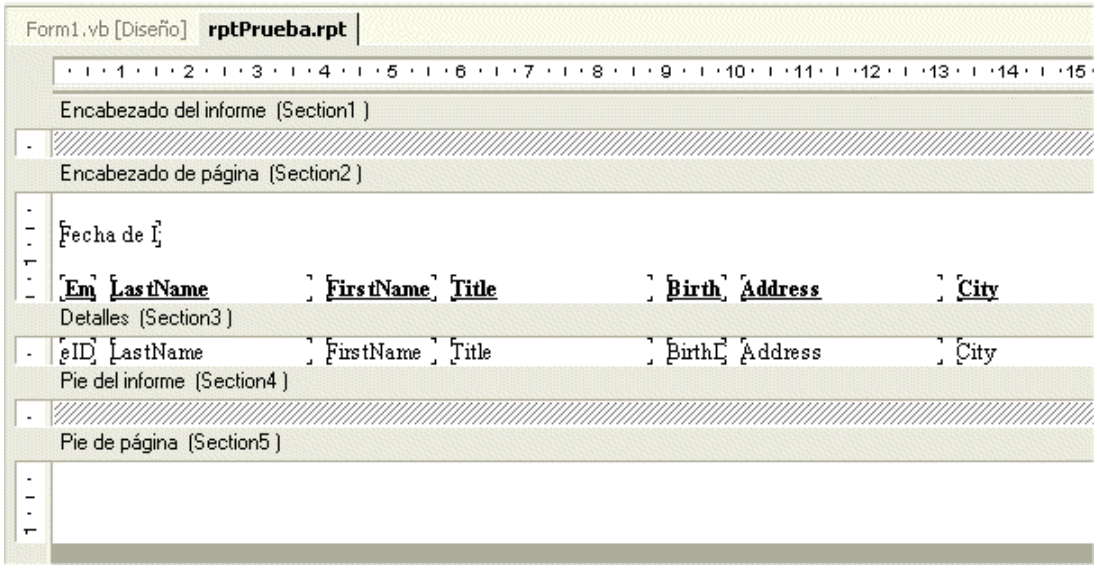


Figura 11. Diseñador de informes de Crystal Reports.

Un informe consta de una serie de secciones, dentro de las cuales se organiza la información. Las secciones básicas que encontraremos habitualmente en todo informe serán las siguientes:

- **Encabezado del informe**. Esta sección se imprime una vez al comienzo de la ejecución del informe, siendo adecuada para insertar en ella información general como título, autor, imágenes, gráficos, etc.

- **Encabezado de página**. Esta sección se imprime una vez por cada página del informe, y en ella podemos situar los títulos de las columnas de los campos de la tabla a imprimir, la fecha del listado, etc.

- **Detalles**. Es la sección más importante, y en ella se sitúan los campos de la tabla que representan los datos del informe, y que se imprimirán tantas veces como registros sea necesario mostrar.

- **Pie del informe**. Se imprime una vez al final del informe, siendo una sección propicia para insertar un total general que muestre el resultado de la suma de columnas numéricas.

- **Pie de página**. Se imprime una vez por página, justamente después de las líneas de detalle. Se trata de una sección muy adecuada para incluir números de página, líneas separadoras, etc.

El contenido actual del informe también puede ser modificado, ya que lo que hemos obtenido con el asistente ha sido un diseño basado en una guía predefinida por CR, y que no siempre coincidirá con el diseño que teníamos previsto realizar para nuestro listado.

Podemos añadir, modificar, quitar campos y otros controles del informe, aunque la confección manual del mismo es una tarea que veremos próximamente, por lo que también más adelante profundizaremos en el uso de este diseñador; el diseño actual por lo tanto será adecuado para el presente ejemplo.

El control CrystalReportViewer

Llegados a este punto hemos finalizado el proceso de diseño del informe. Como habrá comprobado el lector, ha sido una tarea muy fácil, ya que el asistente ha realizado todo el trabajo por nosotros.

Sin embargo ahora necesitamos que al ejecutar nuestra aplicación, podamos imprimir el informe, o lo que es mejor, disponer de la posibilidad de previsualizarlo antes de su impresión definitiva en papel.

Para este fin contamos con el control CrystalReportViewer, que como indica su nombre, se trata de un visualizador de informes que nos permitirá mostrar el listado en un formulario, para su comprobación por parte del usuario. La Figura 12 muestra el icono de este control en el Cuadro de herramientas del IDE.

Figura 12. Control CrystalReportViewer en el Cuadro de herramientas.

Al igual que hacemos con cualquier otro control Windows, insertaremos una copia de este control en el formulario de nuestro proyecto, y le daremos el nombre crvInforme. Para que el informe se muestre con un mayor tamaño, asignaremos a la propiedad Dock de este control el valor Fill. Ver la Figura 13.

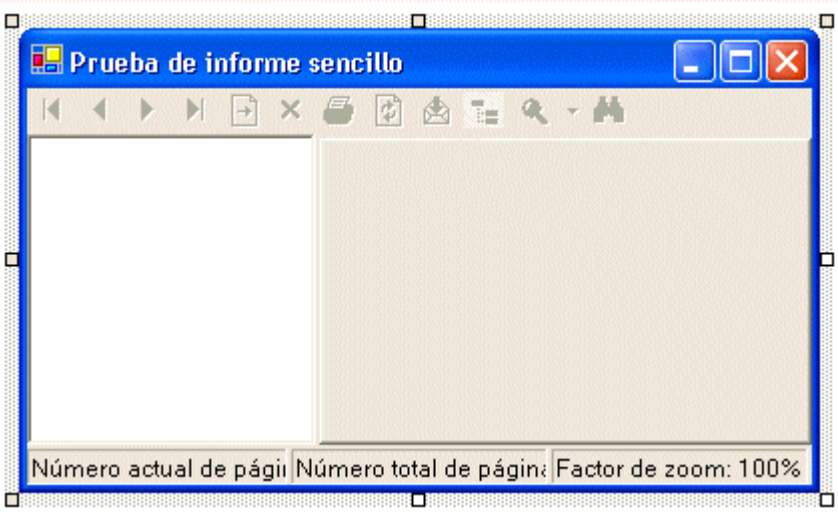


Figura 13. Formulario con control CrystalReportViewer.

Por último, mediante la propiedad ReportSource del control, asignaremos una cadena con la ruta y el nombre del archivo correspondiente al informe que acabamos de crear, para llevar a cabo su visualización. Esta tarea la podemos hacer a través de la ventana de propiedades del control, utilizando una caja de navegación de directorios y archivos, o bien por código. Para este ejemplo optaremos por utilizar código, y en el evento Load( ) del formulario, escribiremos la instrucción que cargará el informe en el visualizador, como vemos en el Código fuente 1.

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    ' cargar el informe en el visualizador
    Me.crvInforme.ReportSource = "E:\PruebasVBNET\RepSencillo\rptPrueba.rpt"
End Sub
```

Código fuente 1

El resultado se muestra en la Figura 14.

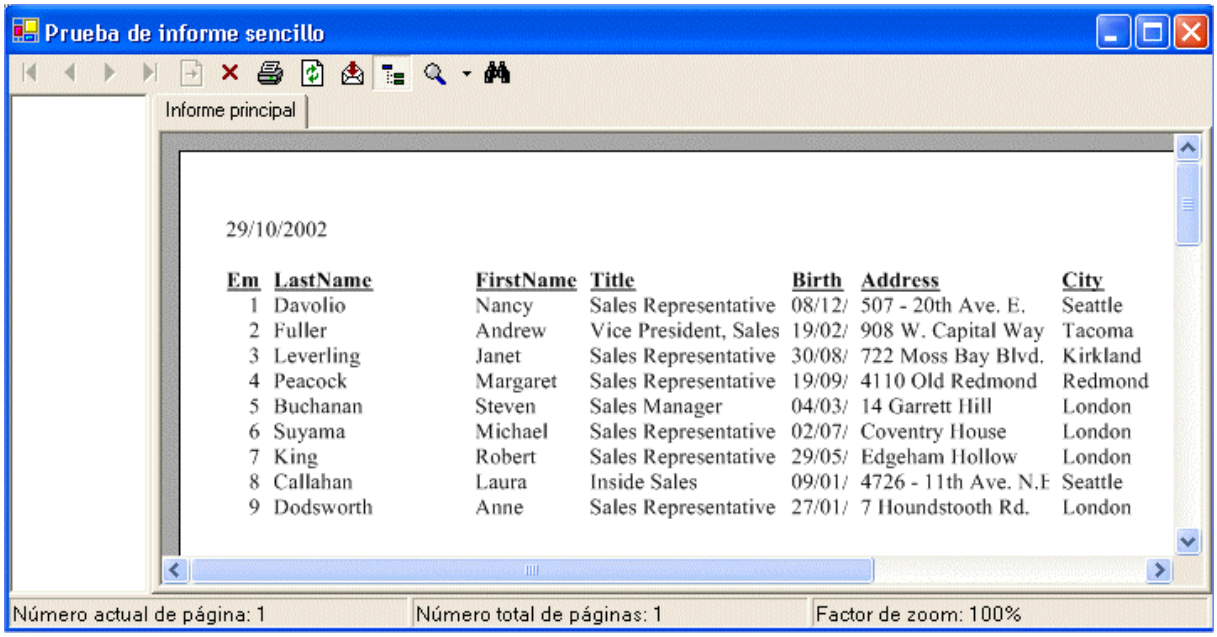


Figura 14. Visualización del informe en el formulario mediante el control CrystalReportViewer.

El control CrystalReportViewer incorpora en su barra de herramientas una serie de funcionalidades estándar, que evitan al programador tener que implementarlas en su aplicación, entre ellas se encuentran la posibilidad de navegar entre las páginas del informe, exportarlo a diversos formatos de documento (PDF, Word, Excel, etc.), buscar texto, etc.

Como punto final al desarrollo y ejecución de nuestro primer informe, una vez que hayamos comprobado que los datos previsualizados son correctos, podemos imprimirlo pulsando el icono de la barra de tareas de este control que tiene forma de impresora, lo que nos mostrará la caja de diálogo estándar de Windows para impresión de documentos.

La clase del informe

Al igual que ocurre con los formularios Windows, los informes de Crystal Reports disponen de un diseñador, para crearlos visualmente, y de una clase, que hereda de la clase ReportClass, y proporciona la funcionalidad necesaria. Para comprobarlo, nos situaremos en la ventana Explorador de soluciones, y haremos clic en el botón *Mostrar todos los archivos*, de su barra de herramientas. De esta forma, además del archivo del informe (el que tiene la extensión .RPT), se visualizará un nuevo archivo dependiente de este, con extensión .VB, que contiene la clase generada por el diseñador. Ver la Figura 15.

En el Código fuente 2 podemos ver un fragmento del código de la clase del informe.

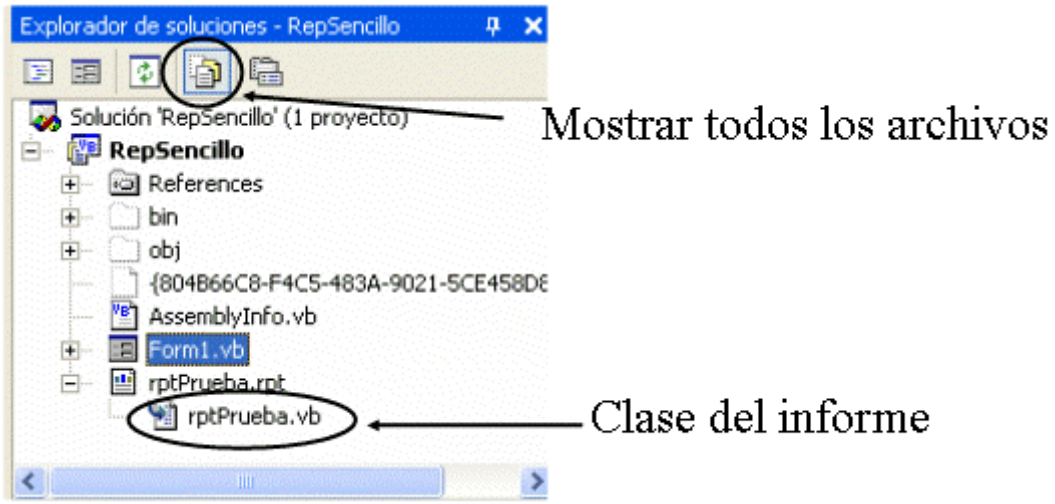


Figura 15. Visualizar el archivo con la clase del informe.

```
Option Strict Off
Option Explicit On

Imports CrystalDecisions.CrystalReports.Engine
Imports CrystalDecisions.ReportSource
Imports CrystalDecisions.Shared
Imports System
Imports System.ComponentModel


Public Class rptPrueba
    Inherits ReportClass

    Public Sub New()
        MyBase.New
    End Sub

    Public Overrides Property ResourceName As [String]
        Get
            Return "rptPrueba.rpt"
        End Get
        Set
            'Do nothing
        End Set
    End Property
'....
```

Código fuente 2

El hecho de que un informe sea internamente una clase, nos proporciona una gran flexibilidad, ya que nos permite crear un objeto a partir de la misma, y manipular el informe a través de sus propiedades y métodos.

Un ejemplo ilustrativo sería en el momento de cargar el informe en el formulario, ya que al control CrystalReportViewer le podemos pasar en su propiedad ReportSource un objeto informe. Para ello, vamos a modificar ligeramente el código del evento Load( ) del formulario, adaptándolo a esta nueva situación. Ver el Código fuente 3.

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    ' antigua versión:
    ' cargar el informe en el visualizador
    'Me.crvInforme.ReportSource = "E:\PruebasVBNET\RepSencillo\rptPrueba.rpt"

    ' nueva versión:
    Dim oInforme As New rptPrueba()
    Me.crvInforme.ReportSource = oInforme

End Sub
```

Código fuente 3

Mediante esta técnica no es necesario preocuparse de la ruta y el archivo que contiene el informe, simplemente hemos de instanciar un objeto de la clase de nuestro informe, y pasarlo al control visualizador.

Tipos de informe

En función de cómo los informes sean manipulados desde un proyecto, podemos clasificarlos en las siguientes categorías.

Informe con establecimiento inflexible de tipos (Strong typed)

Todo informe que constituya parte de un proyecto, es decir, que aparezca al igual que lo hace en el Explorador de soluciones un formulario, clase, etc., se dice que es un informe con *establecimiento inflexible de tipos* o *strong typed*. Este tipo de informe, como hemos explicado en el apartado anterior, es manejable dentro del proyecto como una clase, pudiendo crear instancias de la misma.

Informe sin tipo (Un-Typed)

Todo informe que no constituya parte de un proyecto, y que sea cargado de forma externa en tiempo de ejecución por la aplicación se dice que es un *informe sin tipo* o *un-typed*.

Para utilizar un informe sin tipo desde una aplicación podemos utilizar los dos modos de trabajo que describimos en los siguientes apartados.

Informe sin tipo cargado desde el control CrystalReportViewer

Esta es la técnica más sencilla, y consiste en asignar a la propiedad ReportSource del control CrystalReportViewer incluido en un formulario, una cadena con la ruta de un archivo .RPT que tenga un informe diseñado con CR. El Código fuente 4 muestra un ejemplo en el que se carga en un formulario un informe que no está contenido entre los elementos de un proyecto de Visual Studio .NET.

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
MyBase.Load
```

```
    ' asignar la ruta del archivo del informe
    ' al control visualizador de Crystal Reports
    Me.crvInforme.ReportSource = "C:\Listados\rptFacturas.rpt"
End Sub
```

Código fuente 4.

Informe sin tipo cargado desde el componente ReportDocument

Para utilizar esta técnica, en primer lugar nos situaremos en el Cuadro de herramientas del IDE y haremos clic en el panel Componentes. Entre los elementos que aparecen como componentes seleccionaremos y arrastraremos al diseñador del formulario el componente ReportDocument. Ver la Figura 16.

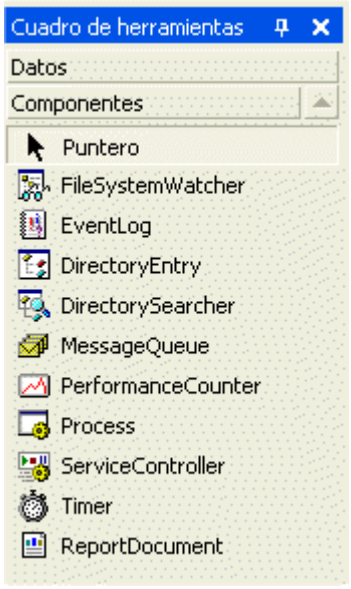


Figura 16. Cuadro de herramientas con el panel Componentes.

La clase ReportDocument representa un objeto informe, y dispone de los miembros para efectuar todas las operaciones de manipulación, de igual forma que si utilizáramos directamente un objeto instanciado de la clase del informe.

Una vez soltado el componente ReportDocument sobre una plantilla de formulario, el motor de informes de CR busca en el proyecto la existencia de informes diseñados, y nos muestra un cuadro de diálogo en el que se solicita asociar el ReportDocument que estamos agregando al formulario con un informe existente, eligiéndolo en una lista desplegable. En el caso de que no vayamos a asociar el componente, como es nuestra situación, seleccionaremos en la lista desplegable el valor *ReportDocument no escrito*, como vemos en la Figura 17.

Tras este paso, quedará en el panel de elementos ocultos del diseñador del formulario, una copia del componente. La Figura 18 muestra un ReportDocument en el formulario al que le hemos dado el nombre rdInforme.

Figura 17. Elegir un informe para asociar al componente ReportDocument.

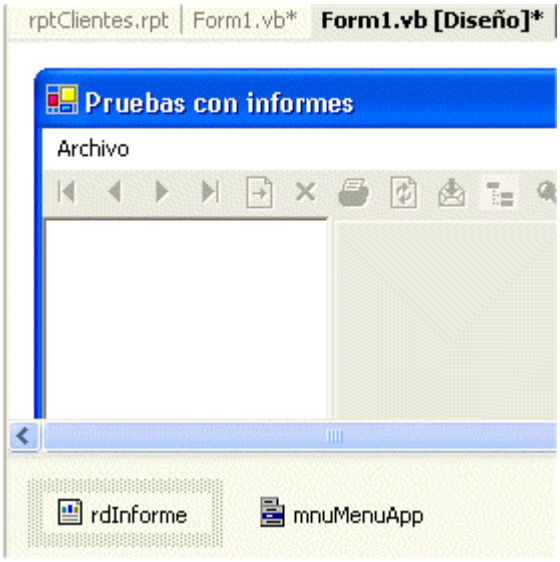


Figura 18. Componente ReportDocument rdInforme añadido en el diseñador del formulario.

Para terminar sólo quedaría escribir el código mediante el cual, el componente ReportDocument de nuestro formulario cargue, a través de su método Load( ) un archivo .RPT que contenga un informe y que se encuentre en cualquier ruta del equipo. Ver el Código fuente 5.

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    ' utilizar un componente ReportDocument para
    ' cargar un informe sin tipo en el visualizador
    ' de informes
    Me.rdInforme.Load("C:\Listados\rptFacturas.rpt")
    Me.crvInforme.ReportSource = Me.rdInforme
End Sub
```

Código fuente 5

El lector pensará con toda probabilidad que esta técnica es más complicada para, a fin de cuentas, obtener el mismo resultado. La ventaja de este modus operandi radica en que, a través del componente ReportDocument, obtenemos una mayor flexibilidad a la hora de manipular el contenido del informe por código.

Creación de un informe manual a partir del diseñador vacío

Para crear un informe con CR no es obligatorio el uso de un asistente que genere los campos como se ha mostrado en el ejemplo anterior. También es posible realizar el diseño del informe partiendo de una plantilla vacía del diseñador, de manera que podamos situar todos sus elementos por nosotros mismos, sin dejar intervenir en este caso a los procesos automáticos del asistente.

La creación de un informe a través del asistente, como hemos comprobado en el ejemplo anterior, es un proceso que simplifica en gran medida el trabajo del programador, evitando que este se ocupe entre otros aspectos del diseño, de la organización y ubicación de los campos en el informe.

Sin embargo, nos encontraremos con situaciones en las que la disposición automática de los campos generada por el asistente, y otros aspectos de su configuración, no se adaptarán a nuestras necesidades. En este tipo de casos deberemos optar por crear nosotros mismos el informe partiendo de la plantilla del diseñador vacía, y establecer la fuente de datos, campos, y demás objetos del listado.

Pongamos como ejemplo un escenario en el que precisamos crear un informe que muestre algunos campos de la tabla Employees, y añadir además, un conjunto de elementos que por defecto no proporciona el asistente.

Vamos para ello a crear un nuevo proyecto con el nombre RepManual (para acceder al proyecto de este ejemplo hacer clic aquí), añadiendo al mismo un informe de la forma explicada en el ejemplo anterior (también con el nombre rptPrueba).

En la ventana Galería de Crystal Report haremos clic en el RadioButton *Como informe en blanco*, para proceder a crearlo totalmente de forma manual. Ver la Figura 19.

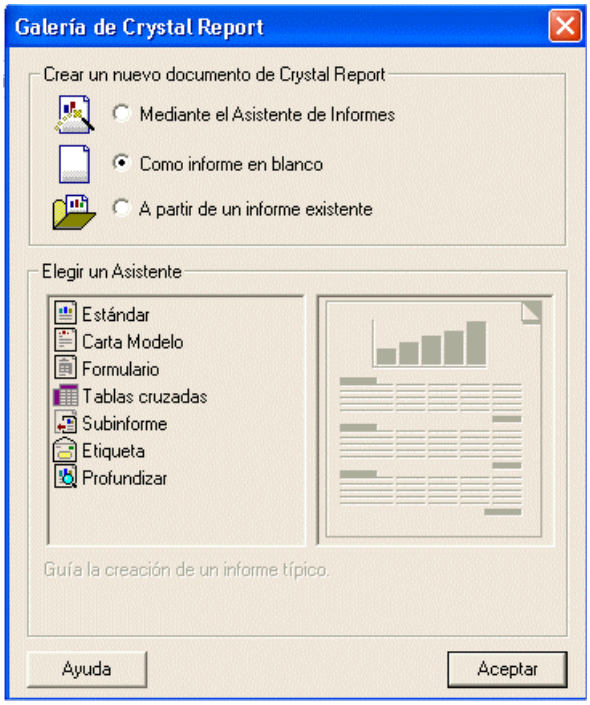


Figura 19. Selección de creación de informe en blanco.

De esta forma se presentará la plantilla del diseñador de informes vacía. Ver la Figura 20.

Figura 20. Diseñador de informes vacío.

Seleccionar y establecer la conexión con un origen de datos

Llegados a este punto haremos clic derecho sobre una zona vacía del diseñador, y elegiremos la opción de menú contextual *Base de datos + Agregar o quitar base de datos*, que nos mostrará un asistente para la conexión con la base de datos igual que el utilizado con el asistente de creación de informes del ejemplo anterior. Tras seleccionar el proveedor OLE DB (ADO) de SQL Server, e introducir los valores de conexión, expandiremos los nodos del panel izquierdo de esta ventana, hasta llegar a la tabla Employees, que añadiremos al panel derecho como tabla a utilizar en el informe. Ver la Figura 21.

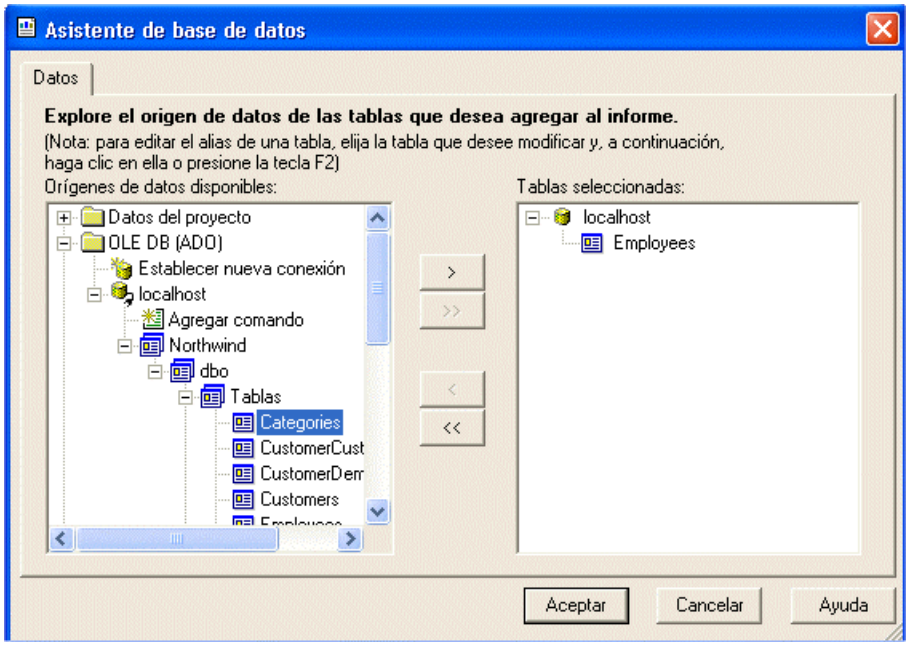


Figura 21. Selección de tabla a mostrar en el informe.

Agregar campos de la tabla al informe

Como paso siguiente expandiremos la ventana Explorador de campos, que se encuentra normalmente en el lateral izquierdo del IDE contraída en forma de solapa, y expandiremos el nodo *Campos de base de datos*, hasta llegar a la tabla y campos que necesitemos. A continuación haremos clic en los campos que vayamos a incluir en el informe, y mediante arrastrar y soltar los llevaremos hasta la sección Detalles del diseñador del informe, añadiéndose el campo y un control de texto como título para el mismo en la sección Encabezado de página. Agregaremos de este modo al informe los campos LastName, FirstName, BirthDate y City. Ver Figura 22.

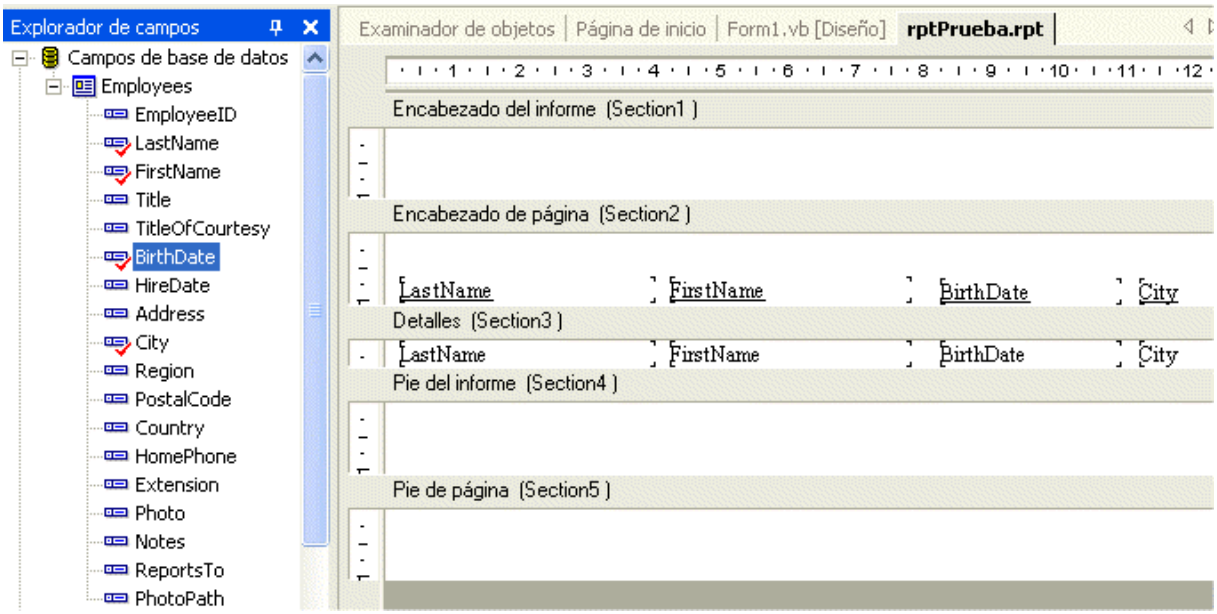


Figura 22. Inserción de campos en el informe.

Manipular objetos en el diseñador del informe

Una vez insertado un campo u otro tipo de control en el diseñador, podemos hacer clic en él y cambiar su tamaño. Igualmente es posible seleccionar un conjunto de objetos del diseñador, y aplicar una operación al mismo tiempo sobre todos ellos.

Por ejemplo, al hacer clic derecho en el objeto de texto correspondiente al título de un campo, podremos modificar el nombre y formato por defecto seleccionando en el menú contextual la opción *Editar objeto de texto*.

Para cambiar de posición los objetos del informe, sólo hemos de hacer clic sobre ellos y moverlos con el ratón o con las teclas de dirección, aunque no podremos situarlos con toda la precisión que necesitemos en el diseñador.

Si necesitamos ajustar de forma más precisa la posición de los objetos del informe, haremos clic derecho sobre el diseñador, eligiendo la opción del menú contextual *Diseñador + Cuadrícula*. Esto nos mostrará una guía para ubicar los controles en el diseñador, igual que la existente en el diseñador de formularios.

No obstante, el espaciado por defecto de puntos en la cuadrícula es muy grande, por lo que para poder ajustarlos mejor, pasaremos a la ventana de propiedades y seleccionaremos rptPrueba, nuestro objeto

Report, modificando la propiedad GridSize a un valor menor del existente, por ejemplo 0,09. Ver la Figura 23.

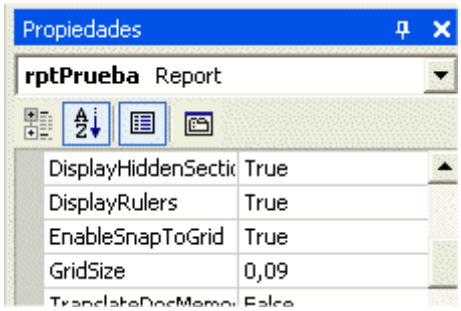


Figura 23. Tamaño de la cuadrícula del diseñador de informes.

El resultado será una trama de puntos mayor, y por consiguiente una mejor organización de los objetos del informe. Ver Figura 24.

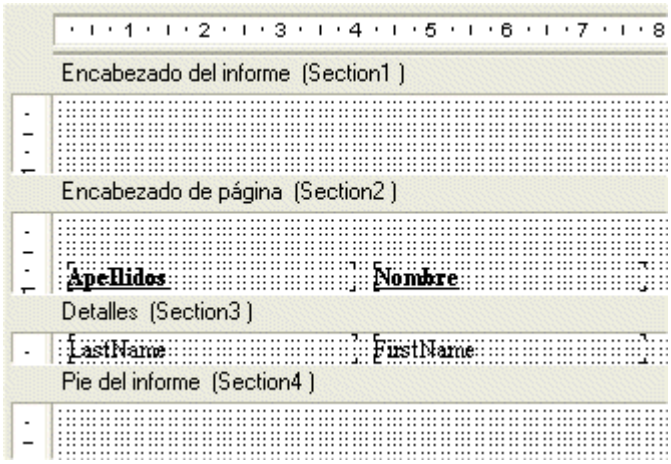


Figura 24. Diseñador del informe mostrando cuadrícula de ajuste de objetos.

Aplicar formato a los campos

Podemos modificar el aspecto o formato que muestra un control del informe haciendo clic derecho sobre el mismo y eligiendo la opción Formato del menú contextual.

En el informe de nuestro ejemplo, vamos a poner en negrita los títulos de los campos utilizando esta ventana de formato. Ver la Figura 25.

En el caso de campos de tipo fecha, se muestran con un formato estándar que en ocasiones no puede ser muy adecuado. El campo BirthDate de nuestro informe visualizaría los valores del modo mostrado en la Figura 26 en el caso de que no modifiquemos su formato.

Supongamos que no queremos mostrar la parte horaria de la fecha, y además necesitamos visualizar el nombre del día de la semana. Vamos a cambiar pues el formato de este campo por uno más adecuado, haciendo clic derecho en el mismo y seleccionando la opción Formato, que nos mostrará la ventana de la Figura 27.

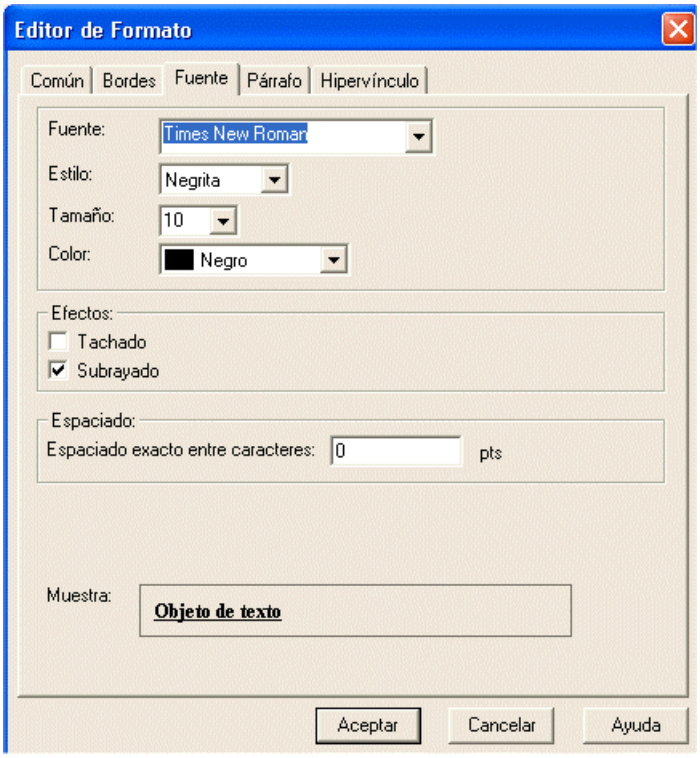


Figura 25. Aplicando formato a un control del informe.

30/08/1963 0:00:00
19/09/1937 0:00:00
04/03/1955 0:00:00
02/07/1963 0:00:00

Figura 26. Fechas mostradas en un informe sin aplicar formato.

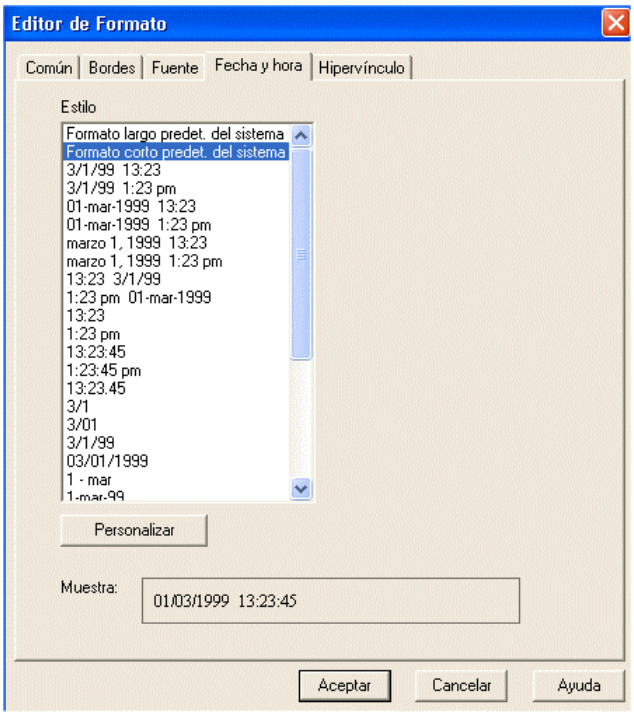


Figura 27. Ventana de formato para campos de fecha.

Como podemos comprobar, disponemos de una extensa lista de formatos predefinidos, aunque en el caso de que no exista ninguno que se adapte a nuestras necesidades, pulsaremos el botón Personalizar, que nos llevará a la ventana *Estilos personalizados*. Una vez en esta ventana, abriremos la lista desplegable Orden, seleccionando el valor Fecha. Ver la Figura 28.

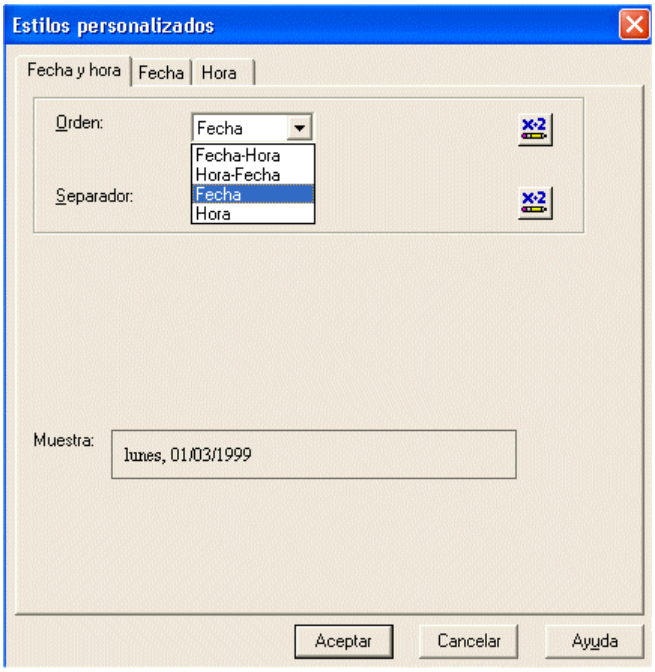


Figura 28. Ventana de estilos personalizados para el formato.

Seguidamente haremos clic en la pestaña Fecha, y construiremos nuestro propio formato, que podremos ir comprobando dinámicamente mediante una fecha de muestra existente en la ventana, que se va actualizando a cada cambio que realizamos. Los botones existentes al lado de cada campo de esta ventana nos permiten crear una fórmula para ese aspecto concreto del formato. La creación de fórmulas será tratada en un apartado posterior Ver la Figura 29.

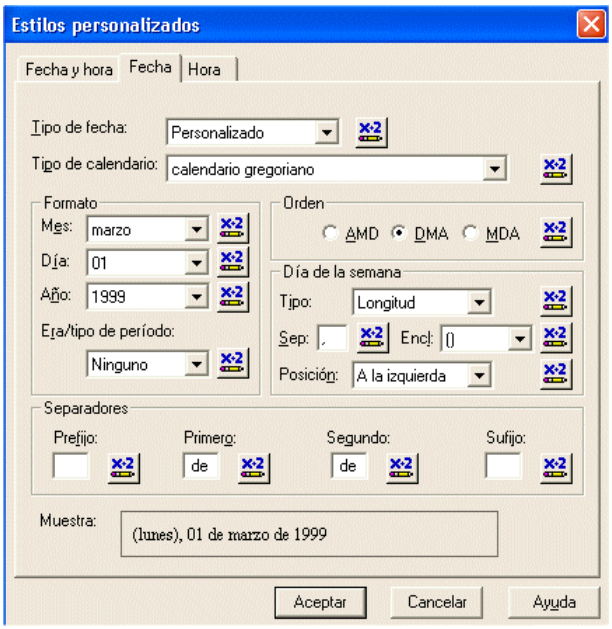


Figura 29. Construir el formato para una fecha.

Con el nuevo formato para el campo de fecha, los valores se mostrarían ahora como indica la Figura 30.

(viernes), 04 de marzo de 1955
(martes), 02 de julio de 1963
(domingo), 29 de mayo de 1960
(jueves), 09 de enero de 1958

Figura 30. Campo de fecha para el informe después de aplicar formato.

Completar el informe con controles adicionales

Llegados al punto actual, nuestro informe proporcionaría toda la información esencial que el usuario necesitaría. No obstante, existen ciertos elementos que sin ser necesarios, su inclusión resulta muy recomendable, ya que dan un valor añadido al informe, y mejoran su aspecto. Los primeros controles en este sentido que vamos a tratar los encontramos en el Cuadro de herramientas, como vemos en la Figura 31.

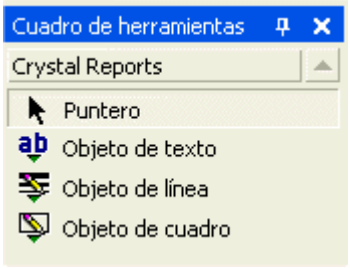


Figura 31. Cuadro de herramientas con controles específicos para Crystal Reports.

De este grupo de controles, situaremos en primer lugar un *Objeto de texto* en la sección Encabezado de informe, escribiendo en su interior una cadena que sirva como título del listado. A continuación, haremos clic derecho sobre este objeto, y en la opción Formato del menú contextual, aumentaremos el tamaño de fuente.

Seguiremos con el control *Objeto de cuadro*, que nos permite dibujar objetos con esta forma en el informe. En nuestro ejemplo dibujaremos un recuadro que encierre al título del informe, y mediante la propiedad LineThickness estableceremos un grueso mayor del que tiene por defecto este objeto.

La misma operación realizaremos con el control *Objeto de línea*, dibujándolo al final de esta sección a modo de línea separadora. La Figura 32 muestra el resultado de la inclusión de estos controles en el informe.

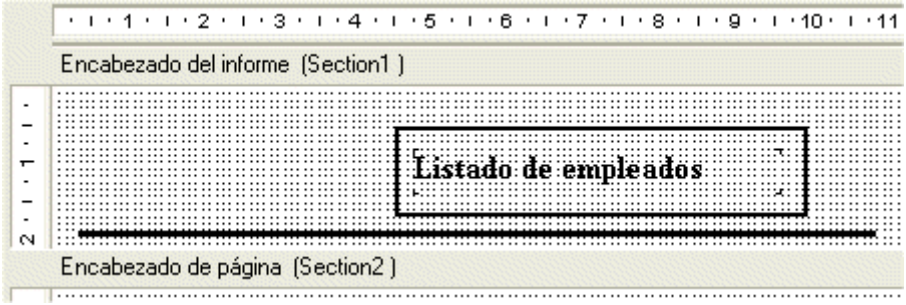


Figura 32. Informe incluyendo controles adicionales.

El siguiente grupo de controles adicionales lo encontramos en la ventana Explorador de campos, dentro del nodo *Campos especiales*. Debido al gran número de esto campos, mostraremos sólo algunos a modo de muestra. Ver la Figura 33.

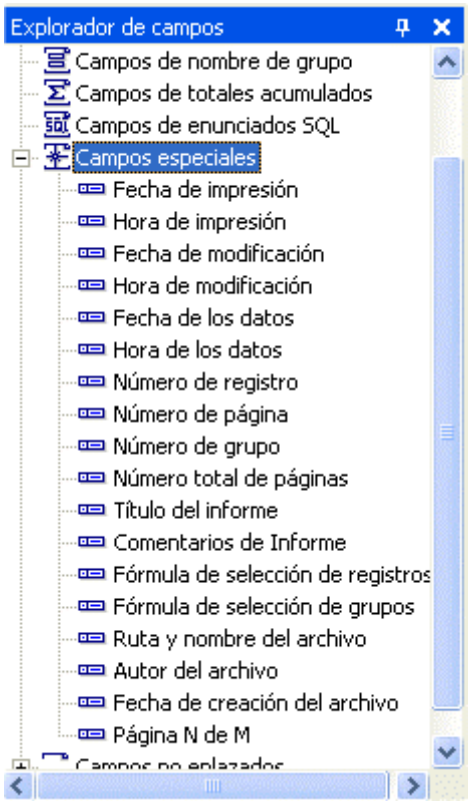


Figura 33. Campos especiales del Explorador de campos.

Fecha y hora del informe

Vamos a insertar en la sección Encabezado de página del informe dos campos: *Fecha de impresión* y *Hora de impresión*, a los que modificaremos el formato en cuanto al tamaño de letra y el mes de la fecha, para que resalten del resto de campos del informe. Añadiremos también un campo de texto a modo de etiqueta informativa, y un objeto de línea bajo todos ellos. El resultado se muestra en la Figura 34.

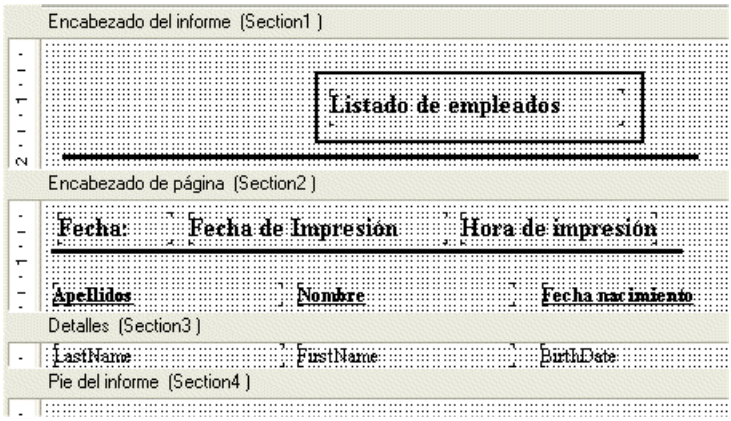


Figura 34. Campos especiales de fecha y hora del informe insertados en el diseñador.

De esta forma, cada vez que se imprima una nueva página, se incluirá la fecha y hora en que se ha generado el listado.

Indicadores de página

A continuación insertaremos en la sección Pie de página un campo *Página N de M*, que nos servirá para mostrar el número de página actual del informe y el número total de páginas que lo componen. Ver la Figura 35.

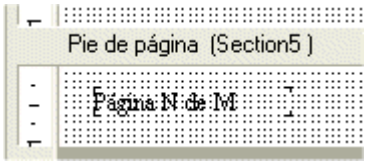


Figura 35. Campo de número de página actual y total de páginas.

El mismo resultado habríamos obtenido empleando por separado los campos *Número de página* y *Número total de páginas*. Sin embargo esta técnica resulta más trabajosa, puesto que se tratan de dos campos separados, y además tendríamos que haber añadido algún campo de texto para mejorar esta información. Ver la Figura 36.

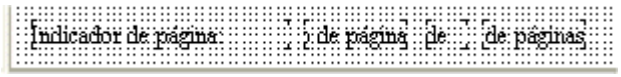


Figura 36. Campos de página actual y total de páginas en el informe.

Imagen

Si queremos añadir un archivo que contenga una imagen, al estilo de un logotipo para el listado, debemos hacer clic derecho sobre el informe y seleccionar la opción del menú contextual *Insertar* + *Imagen*, que abrirá un cuadro de diálogo para seleccionar el archivo a insertar en el diseñador.

En nuestro ejemplo hemos situado una pequeña imagen en la sección de Encabezado de informe, como muestra la Figura 37.

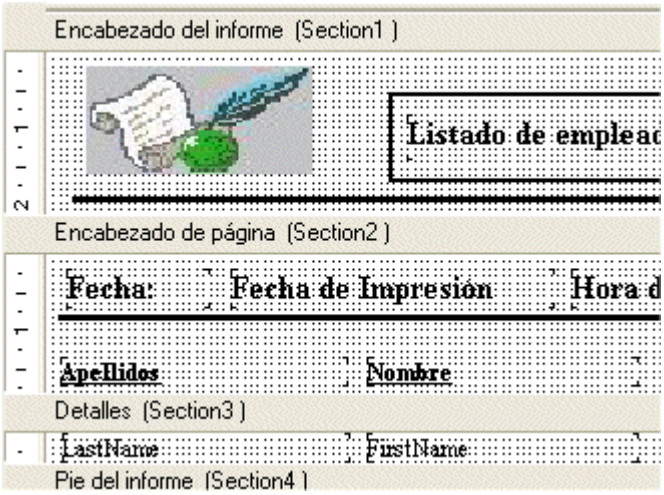


Figura 37. Informe incorporando imagen.

Tras incluir este último elemento, ya sólo queda añadir un control CrystalReportViewer con el nombre crvInforme al formulario del proyecto, y escribir en su evento Load( ) el código que realiza la carga del informe en este control, para poder previsualizarlo en la ventana de la aplicación. Ver el Código fuente 6.

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    Dim oRptPrueba As New rptPrueba()
    Me.crvInforme.ReportSource = oRptPrueba
End Sub
```

Código fuente 6

Al ejecutar el proyecto, el informe mostrará un aspecto similar al de la Figura 38.

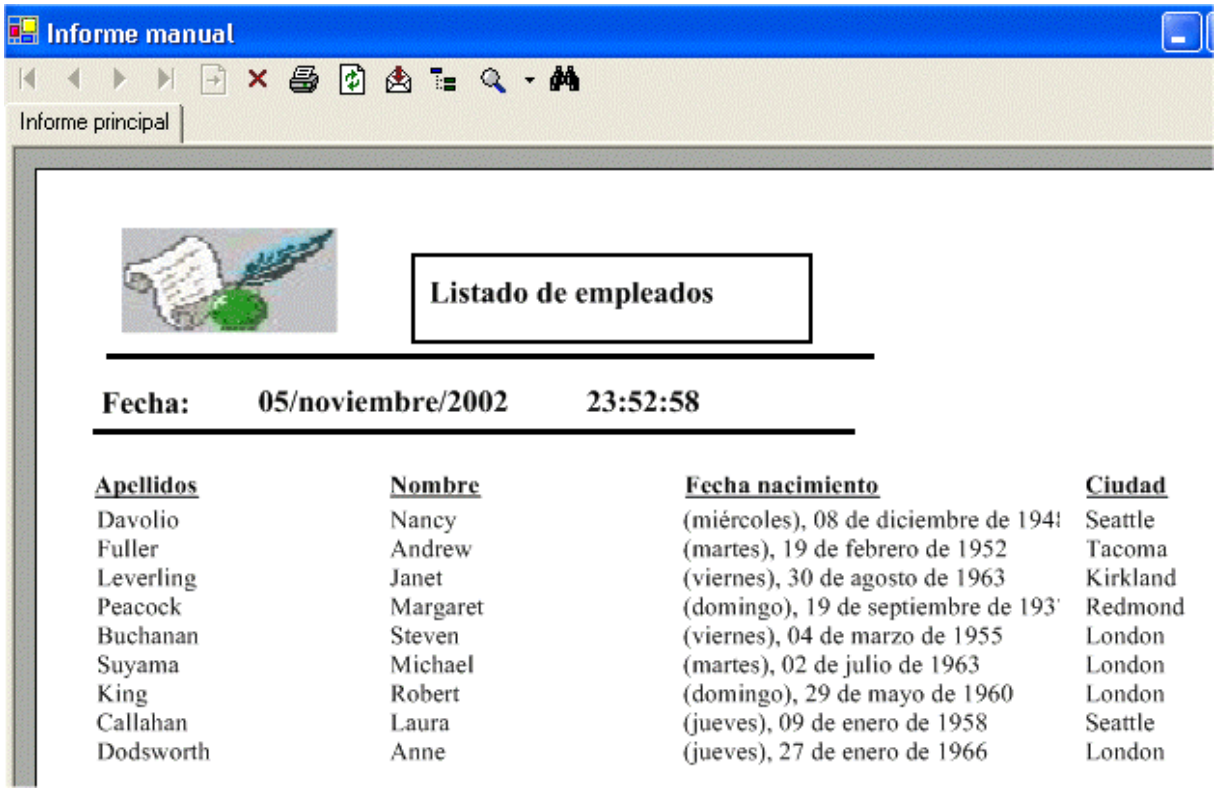


Figura 38. Informe creado de forma manual, con campos adicionales.

Informe con datos ordenados

Podemos mostrar las filas del informe ordenadas por uno o más campos de la tabla que usamos como origen de los datos.

Para lograr este objetivo vamos a crear un nuevo proyecto con el nombre RepOrdenado (para acceder al proyecto de este ejemplo hacer clic aquí), al que añadiremos un informe con el nombre rptPrueba, igual que anteriores ejemplos.

En el asistente de creación del informe seleccionaremos la tabla Customers como origen de los datos, y añadiremos al listado los campos CustomerID, CompanyName, City y Country. Tras este paso podemos finalizar el asistente de informes.

A continuación haremos clic derecho en el diseñador seleccionando del menú contextual la opción *Informe + Ordenar registros*, que mostrará la ventana Dirección de ordenamiento de registros, en la que seleccionaremos de su panel Campos disponibles, el campo Customers.Country, que trasladaremos al panel *Ordenar campos*, en el que además podemos establecer que el modo de ordenación sea ascendente o descendente. Finalmente pulsaremos Aceptar para completar esta fase de creación de orden. Ver la Figura 39.

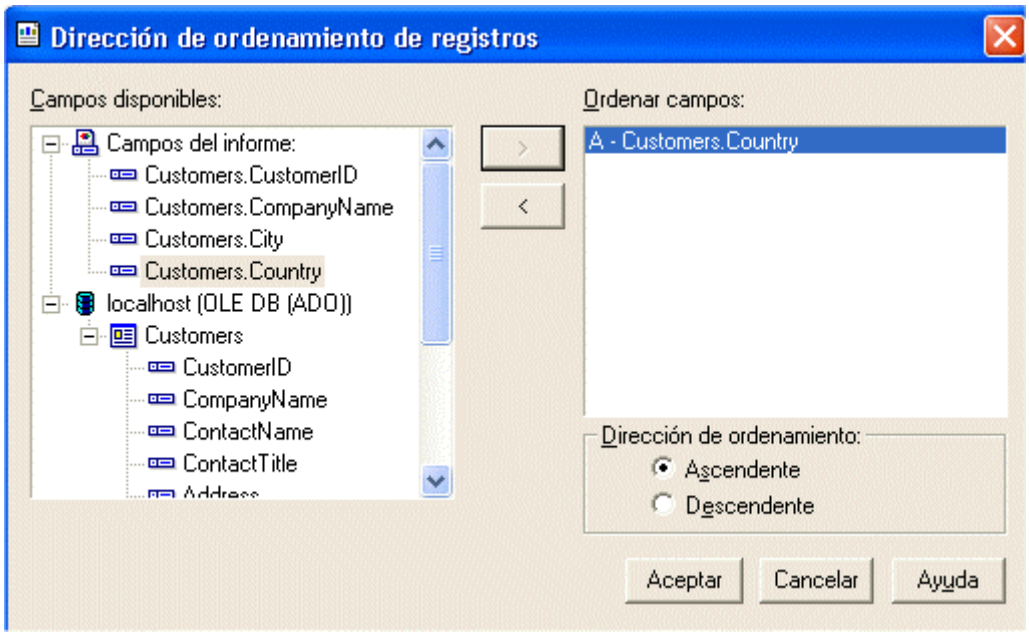
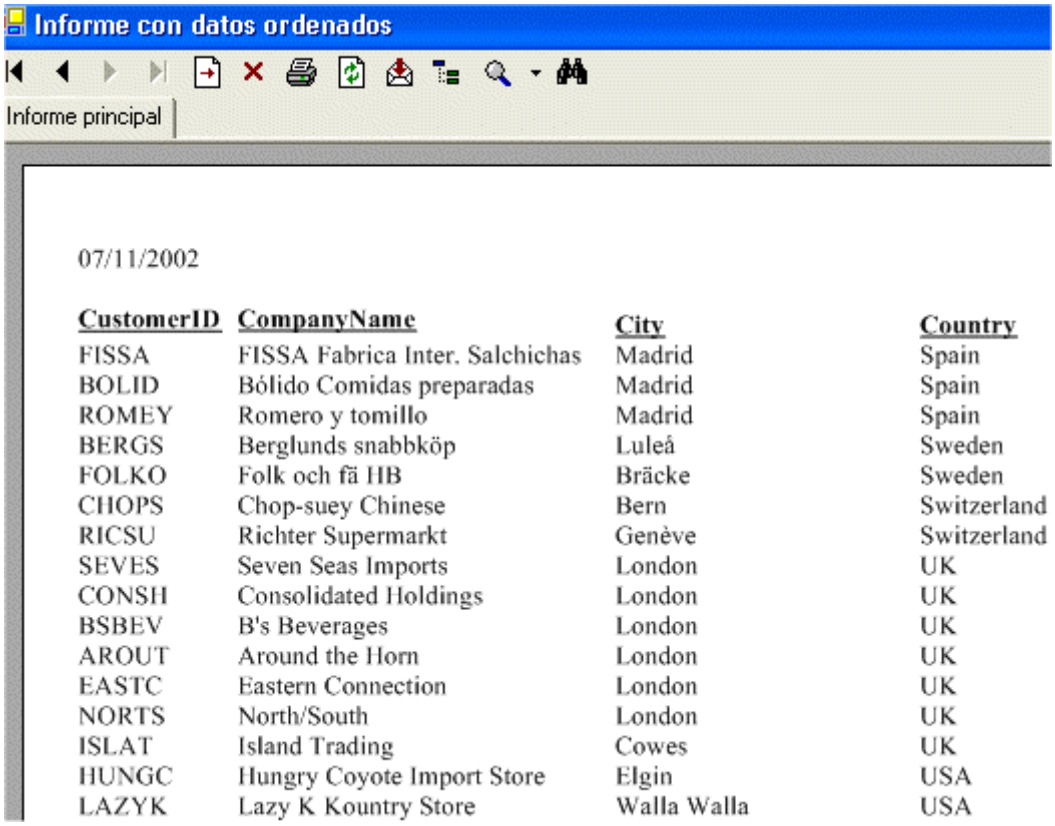


Figura 39. Ventana de selección del campo de ordenación de registros.

Por simplicidad, en este ejemplo ordenamos los registros por un único campo, pero como puede comprobar el lector, podemos añadir varios campos de ordenación al informe. También es importante tener en cuenta, que no es necesario que el campo(s) que forma parte del ordenamiento se incluya en el diseñador del informe.

Tan sólo restaría ahora añadir el control visualizador para el informe en el formulario, y escribir el código para cargar el informe en el control, como ya hemos visto en pasados ejemplos. La Figura 40 muestra el programa en ejecución con el informe ordenado por el campo Country.



CustomerID	CompanyName	City	Country
FISSA	FISSA Fabrica Inter. Salchichas	Madrid	Spain
BOLID	Bólido Comidas preparadas	Madrid	Spain
ROMEY	Romero y tomillo	Madrid	Spain
BERGS	Berglunds snabbköp	Luleå	Sweden
FOLKO	Folk och fä HB	Bräcke	Sweden
CHOPS	Chop-suey Chinese	Bern	Switzerland
RICSU	Richter Supermarkt	Genève	Switzerland
SEVES	Seven Seas Imports	London	UK
CONSH	Consolidated Holdings	London	UK
BSBEV	B's Beverages	London	UK
AROUT	Around the Horn	London	UK
EASTC	Eastern Connection	London	UK
NORTS	North/South	London	UK
ISLAT	Island Trading	Cowes	UK
HUNGC	Hungry Coyote Import Store	Elgin	USA
LAZYK	Lazy K Kountry Store	Walla Walla	USA

Figura 40. Informe con datos ordenados por un campo.

Seleccionar los datos a mostrar en el informe

Los informes de los anteriores ejemplos se han configurado de modo que muestren la totalidad de los registros de la tabla de la base de datos que se usa como origen de la información. No obstante, en la gran mayoría de las aplicaciones también precisaremos la creación de listados que visualicen sólo un rango del total de las filas que contiene una tabla.

Para ilustrar esta situación crearemos un proyecto con el nombre RepSeleccionar (para acceder al proyecto de este ejemplo hacer clic aquí), y como en ejemplos anteriores, le añadiremos un informe con el nombre rptPrueba.

En el asistente para la creación del informe, utilizaremos la tabla Customers, añadiendo al diseñador los campos CustomerID, CompanyName, City y Country.

Tras la selección de campos, pasaremos directamente a la pestaña Seleccionar del asistente. En este paso elegiremos el campo Country, pasándolo al panel *Campos seleccionados*, y estableciendo que el criterio de ordenación sea igual al valor Spain. Ver la Figura 41.

De esta forma, al ejecutar el programa, en el listado sólo se mostrarán los registros que cumplan la condición antes establecida. Ver la Figura 42.

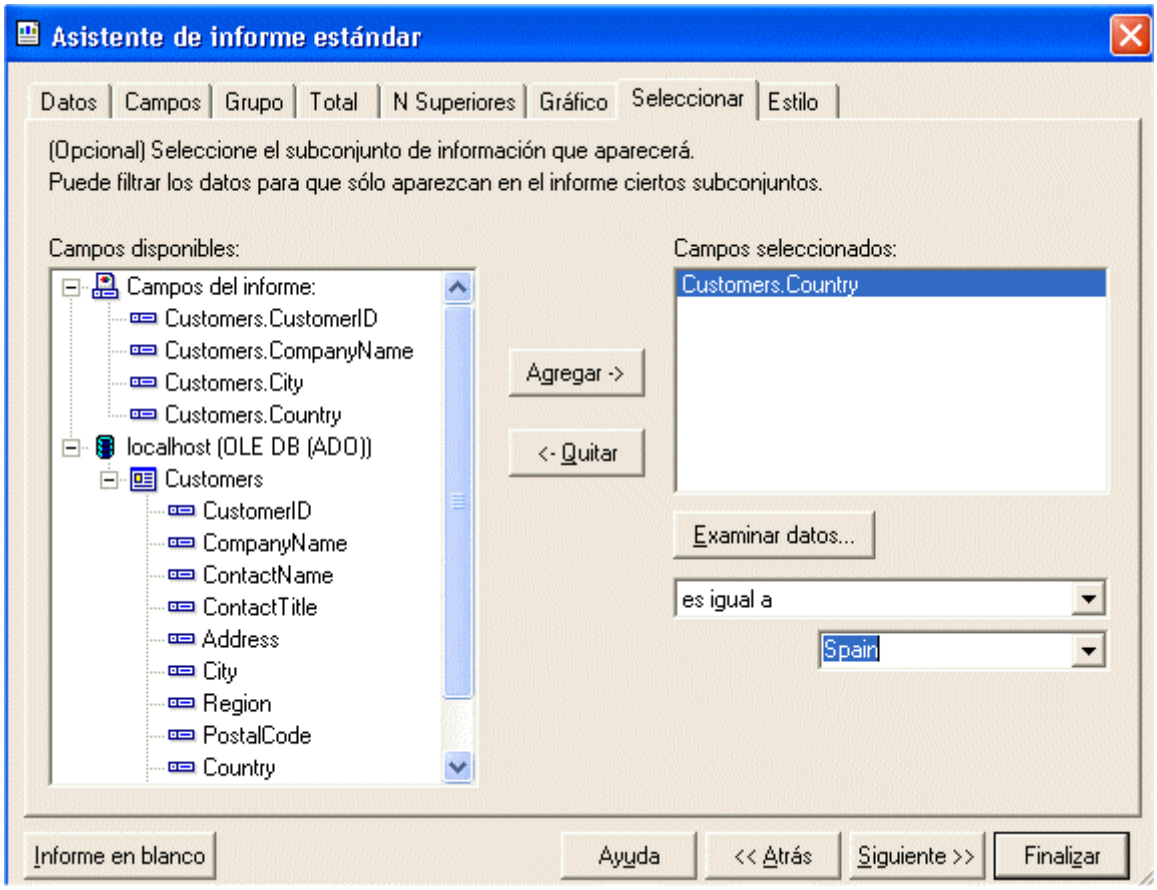


Figura 41. Selección de valores a mostrar en el informe.

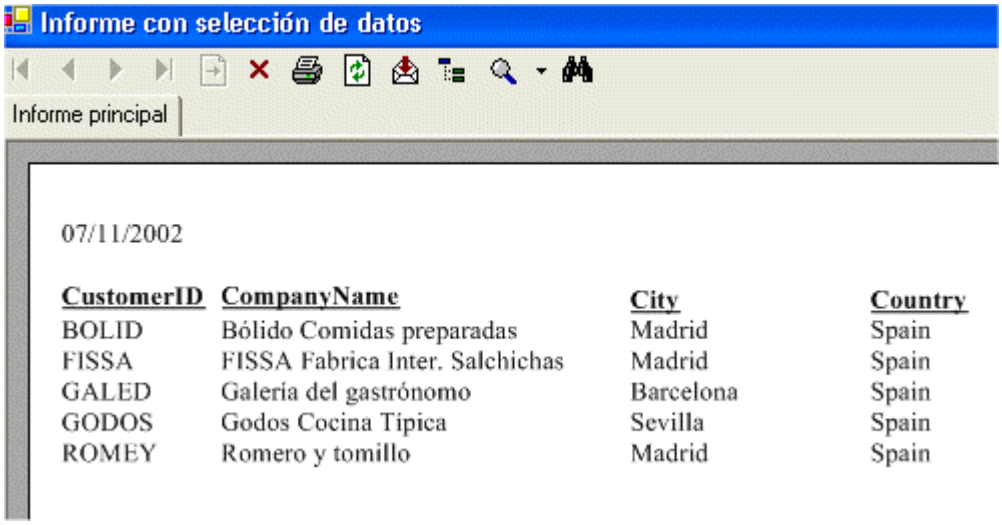


Figura 42. Ejecución del informe con selección de datos.

Si desde el diseñador necesitamos cambiar el criterio de selección debemos hacer clic derecho en el informe y seleccionar del menú contextual la opción *Informe + Asistente de selección*, que abrirá una ventana con la selección actual, permitiéndonos modificarla. Ver Figura 43.

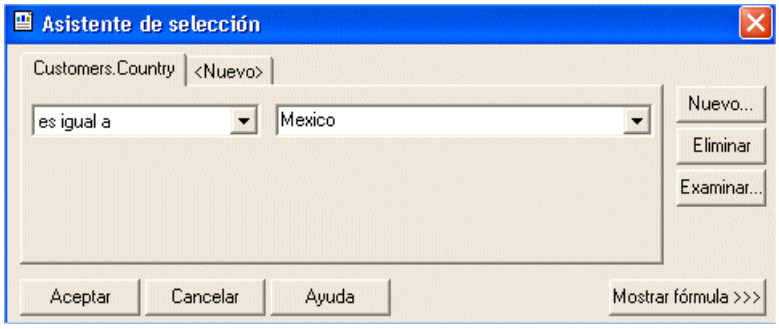


Figura 43. Ventana para el cambio del criterio de selección de registros del informe.

Creación de grupos de registros en el informe

Al diseñar un informe es posible organizar los registros de tal manera que los campos que tienen un mismo valor sean situados juntos, formando un grupo común.

Para ilustrar este aspecto de Crystal Reports crearemos un proyecto con el nombre RepGrupo (para acceder al proyecto de este ejemplo hacer clic aquí), añadiendo un informe de ejemplo, en el que mostraremos el contenido de la tabla Customers.

En el asistente de creación del informe, tras la selección de los campos de la tabla a mostrar, pasaremos a la pestaña Grupo. Del panel Campos disponibles tomaremos el campo Country, pasándolo al panel Agrupar por, de forma que la agrupación de registros sea realizada por dicho campo. Ver Figura 44.

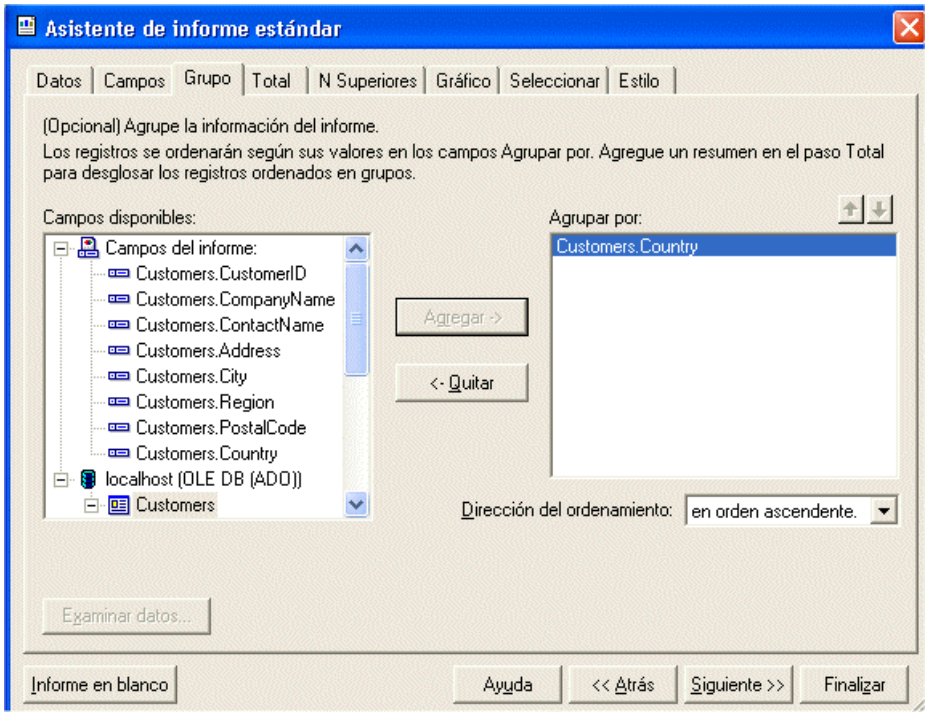


Figura 44. Creación de un grupo para el informe.

En este paso del asistente pulsaremos el botón Finalizar para generar el informe. Observe el lector que este informe incluye una nueva sección con su propio encabezado y pie, correspondiente al grupo por

el que se organizarán los registros. En el título de esta sección se incluye el número de grupo (#1 en este ejemplo), más el nombre del campo que se utiliza para crear la agrupación. Por otra parte, dentro de cada sección de este grupo (encabezado y pie) se ha añadido el campo del nombre del grupo que se acaba de crear. Ver Figura 45.

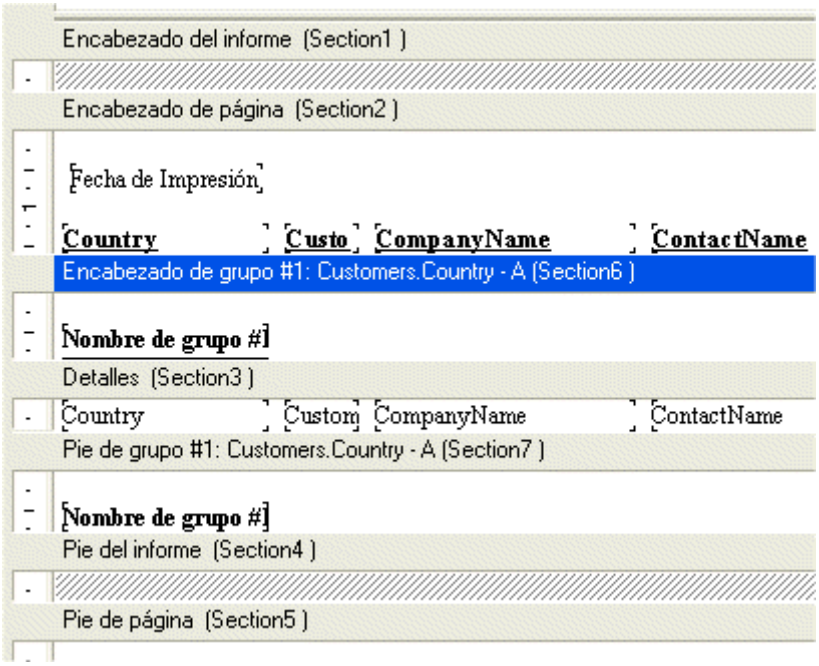


Figura 45. Diseñador de informe incluyendo un grupo.

Los campos de grupo generados también están disponibles en la ventana Explorador de campos, nodo *Campos de nombre de grupo*, como vemos en la Figura 46.

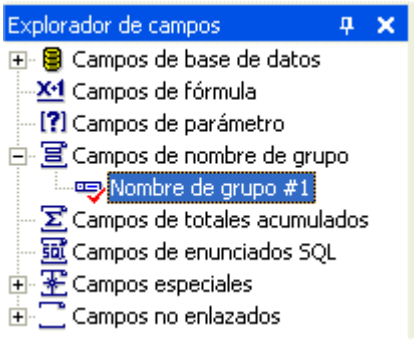


Figura 46. Nodo Campos de nombre de grupo en el Explorador de campos del diseñador de CR.

Debemos tener en cuenta que el resultado de este diseño es modificable, por lo que si la distribución de campos generada por el asistente no se adapta a nuestras necesidades, podemos cambiarlos libremente. Para simplificar este ejemplo, y como ya sabemos modificar los campos contenidos en un informe, dejaremos la distribución tal y como la crea el asistente. Veamos en la Figura 47 el resultado del informe en ejecución, al ser visualizado en un formulario.

Figura 47. Ejecución del informe con registros agrupados.

En los anteriores ejemplos, el lector seguramente se habrá preguntado qué funcionalidad tenía el panel localizado a la izquierda del informe en el control CrystalReportViewer. Como puede comprobar en este ejemplo, su utilidad se manifiesta en informes con registros agrupados, ya que de una manera muy fácil y rápida nos permite desplazarnos a un grupo en concreto al hacer clic en su nombre dentro de este panel. Pulsando el botón de la barra de herramientas *Mostrar/ocultar árbol de grupos* (tercero comenzando por la derecha) dentro de este control, podremos ocultarlo dejando más espacio para el informe en el formulario.

Si necesitamos modificar el campo por el que se produce la agrupación, debemos hacer clic derecho sobre el campo del grupo desde la ventana Explorador de campos, eligiendo la opción de menú contextual *Asistente de Cambio de Grupo*, que nos mostrará la ventana *Cambiar grupo*, con el campo de agrupamiento actual. Pulsando el botón Opciones de esta ventana, aparecerá a su vez la ventana *Cambiar opciones de grupo*, en la que podremos cambiar el campo de agrupamiento, junto con algunas opciones adicionales. Ver la Figura 48.

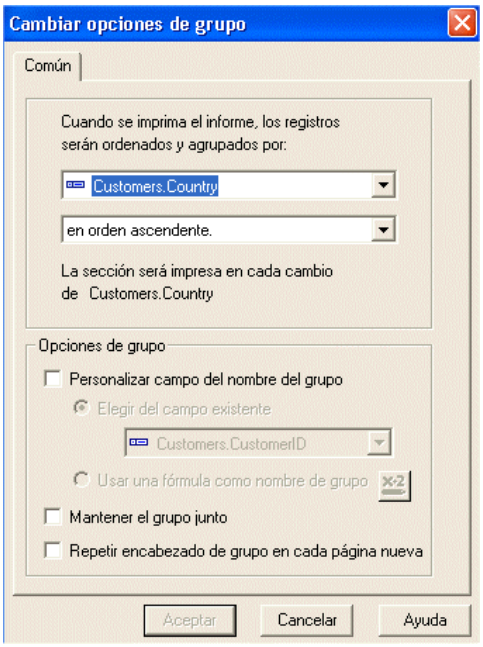


Figura 48. Ventana para cambiar el campo del grupo.

Incluir manualmente el grupo en el informe

Si creamos un informe con el asistente sin incluir un grupo, es posible añadirlo posteriormente haciendo clic derecho en el diseñador, seleccionando en el menú contextual la opción *Insertar + Grupo*, que mostrará la ventana Insertar grupo, en la que podremos elegir el campo para agrupar. Ver la Figura 49.

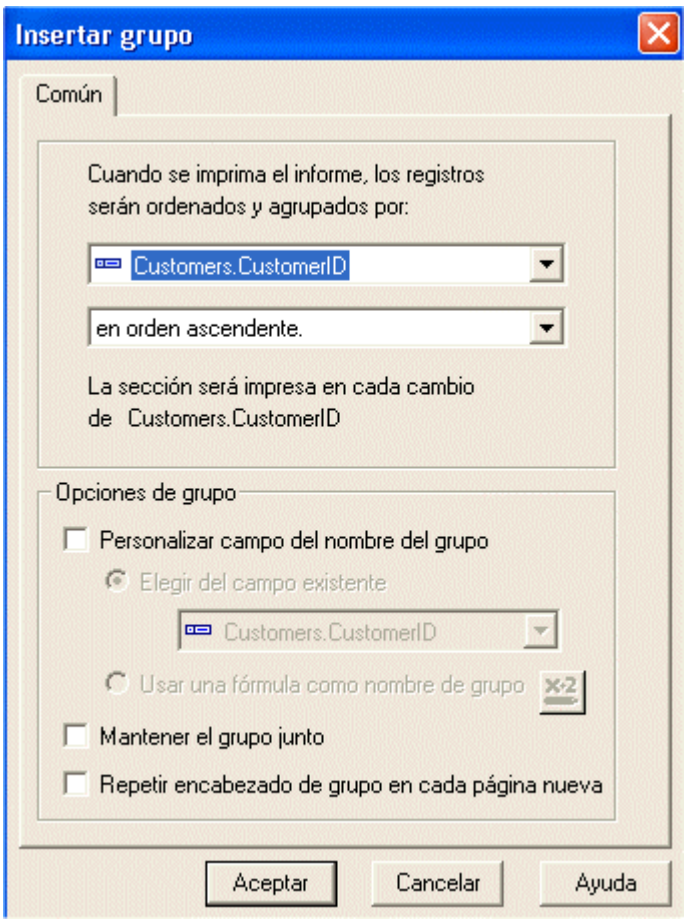


Figura 49. Ventana para insertar un grupo en el informe.

Informe con tablas vinculadas

La gran mayoría de aplicaciones tienen tablas en su base de datos que se encuentran relacionadas en un estilo maestro-detalle; el ejemplo más claro de esto lo constituyen las típicas tablas de facturas y líneas de factura.

Para estas tablas necesitaremos crear un informe que muestre de modo jerárquico o combinado, por cada registro de la tabla facturas, los registros dependientes en la tabla líneas de factura.

Gracias a Crystal Reports, este escenario de trabajo es fácilmente resoluble, bien a través del asistente de creación de informes, o directamente de forma manual en el diseñador.

Para mostrar ambos tipos de abordaje de la situación, crearemos un proyecto con el nombre RepTabVinculadas (para acceder al proyecto de este ejemplo hacer clic aquí).

Debido a que en este proyecto vamos a crear dos informes, añadiremos al formulario del proyecto un menú con dos opciones, que tengan los nombres Asistente y Manual respectivamente, y que nos permita cada una, cargar uno de los informes en el control CrystalReportViewer del formulario.

Como tablas de ejemplo de la base de datos Northwind, utilizaremos Orders y Order Details, que corresponden a una tabla de facturas y líneas de factura respectivamente.

Tablas vinculadas con el asistente

En primer lugar, añadiremos al proyecto un informe con el nombre rptAsistente, en el que como su nombre indica utilizaremos totalmente el asistente para su confección.

Tras configurar la conexión al servidor SQL Server, en el paso de selección de tablas tomaremos las antes mencionadas como vemos en la Figura 50.

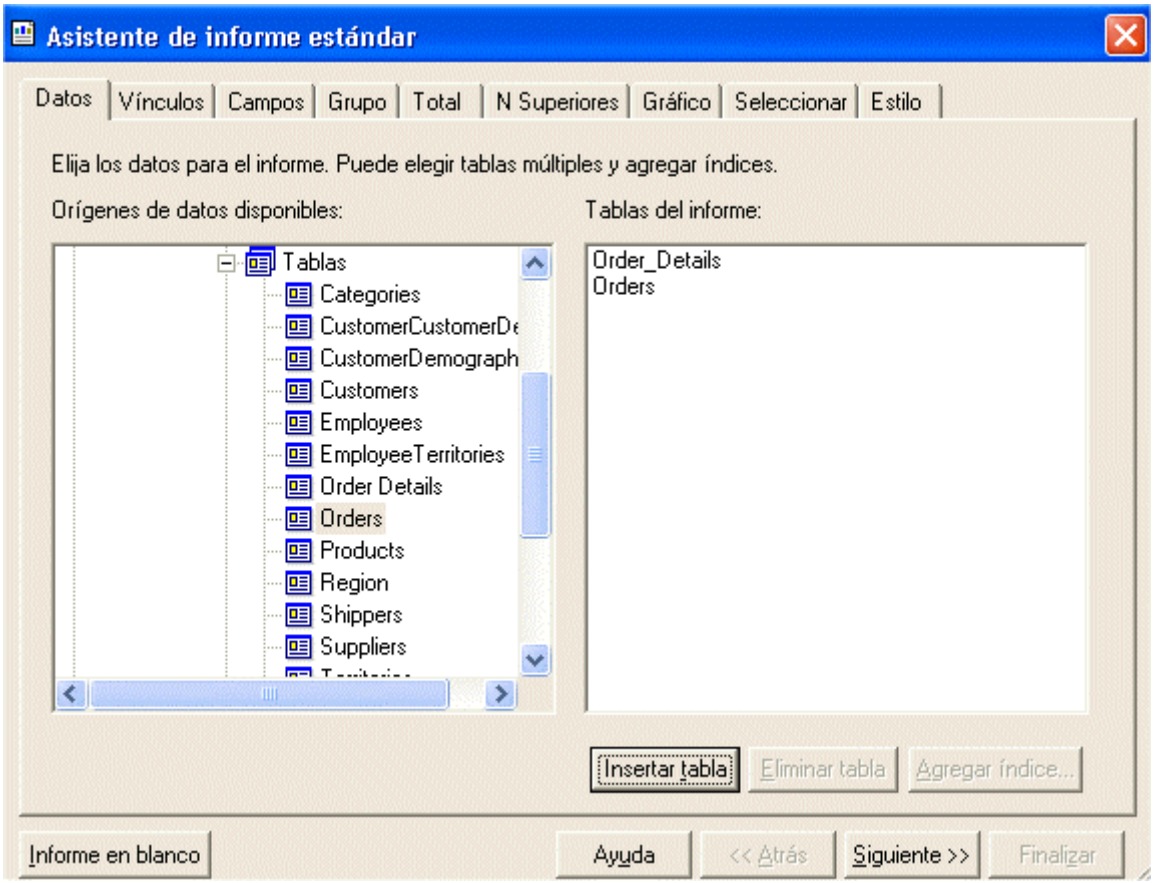


Figura 50. Selección de tablas para informe con datos vinculados.

Observemos en este paso un detalle importante: al añadir más de una tabla, el motor de generación de informes de CR busca la existencia de vínculos entre las tablas por campos comunes, y en caso afirmativo, añade al asistente una pestaña adicional con el nombre Vínculos, que será el paso al que vayamos a continuación. En dicho paso en principio no tenemos que tomar ninguna acción, ya que el propio asistente ha creado automáticamente el vínculo entre ambas tablas por el campo clave común. Si este no fuera el caso, podemos quitar el vínculo automático y crear nosotros uno, o utilizar cualquiera de los elementos de configuración del vínculo que se muestran en la Figura 51.

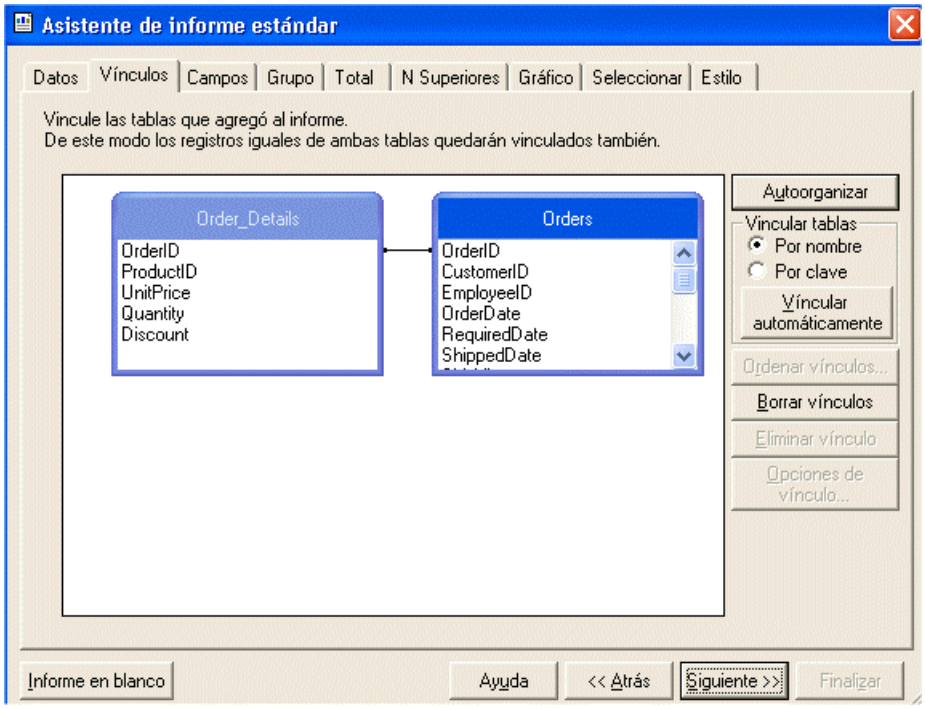


Figura 51. Creación de vínculos entre tablas del informe.

Seguiremos con la selección de campos de las dos tablas que mostraremos en el informe y que vemos en la Figura 52.

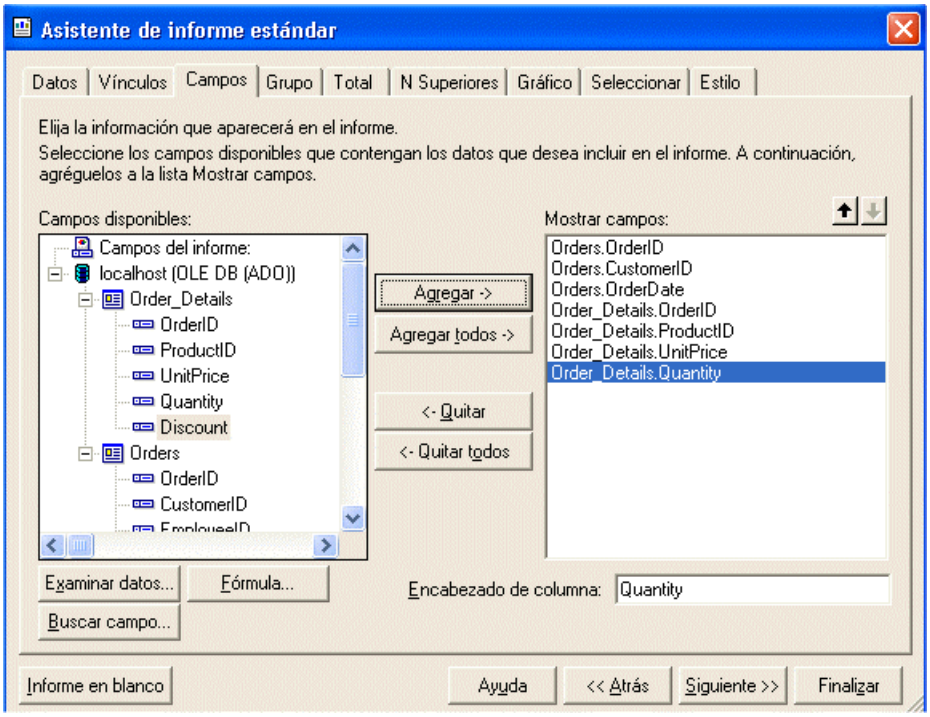


Figura 52. Selección de campos para el informe de las tablas vinculadas.

Para terminar el asistente, en el paso Grupo seleccionaremos el campo OrderID de la tabla Orders, como el campo por el que se agruparán los registros del informe. Ver la Figura 53.

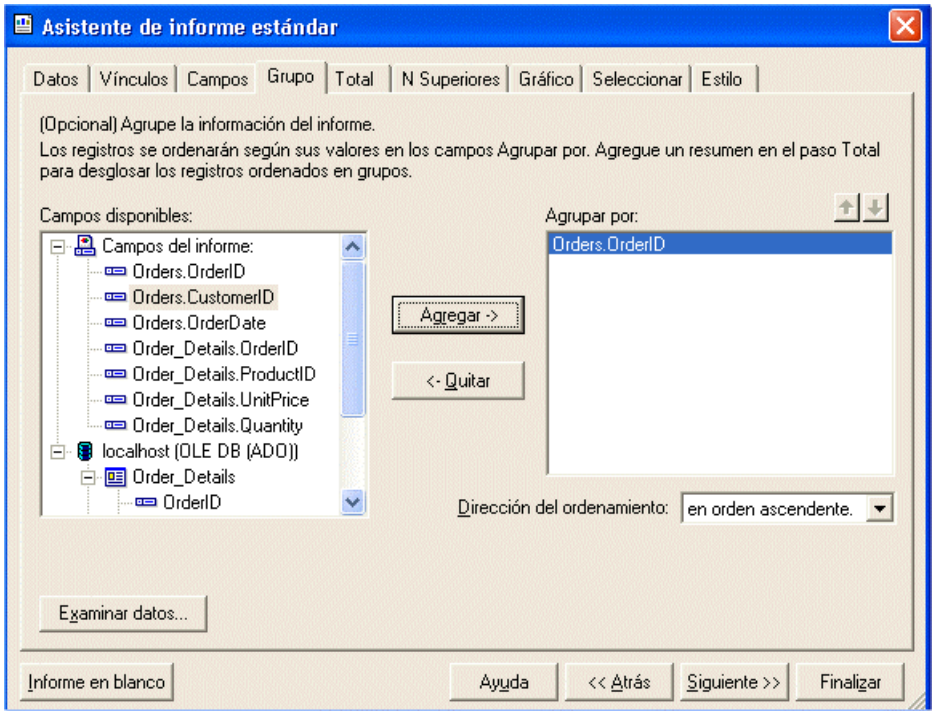


Figura 53. Selección del campo para agrupar registros en el informe.

Al pulsar Finalizar, el asistente producirá el informe que vemos en la Figura 54, en el cual, además de los campos y grupos normales, se incluyen una serie de totales generales y subtotales por grupo creados automáticamente. La creación de totales y subtotales será comentada en un próximo apartado.

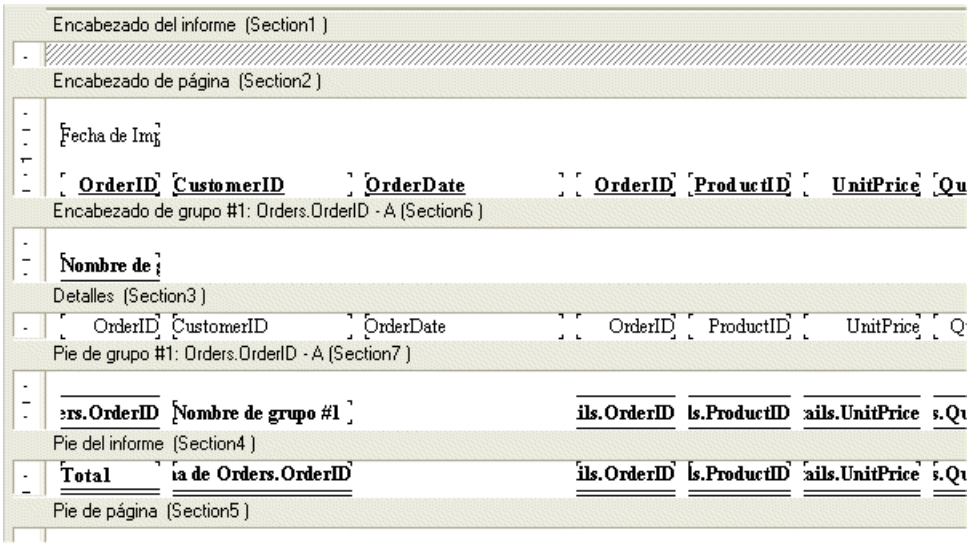


Figura 54. Diseñador de informe con campos de tablas vinculadas.

Para visualizar el informe, en el evento Click( ) de uno de los controles MenuItem del formulario escribiremos las líneas del Código fuente 7.

```
Private Sub mnuAsistente_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mnuAsistente.Click
```

```
    Dim oRptAsistente As New rptAsistente()
    Me.crvInforme.ReportSource = oRptAsistente
End Sub
```

Código fuente 7

El resultado cuando seleccionemos la opción de menú *Archivo + Asistente*, será el mostrado en la Figura 55.

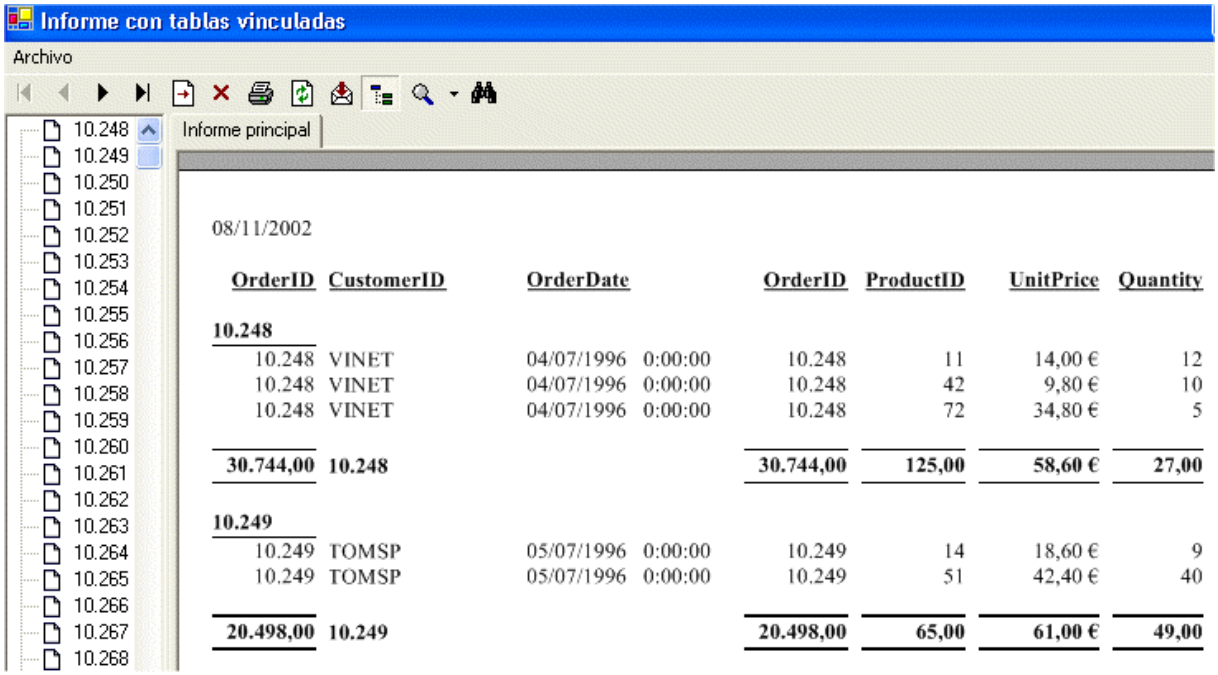


Figura 55. Informe con tablas vinculadas estilo maestro-detalle.

La generación automática de este informe tiene como inconveniente el hecho de que no sitúa los campos jerárquicamente, es decir, en cada grupo sería más recomendable que apareciera una línea con los datos del registro maestro de la tabla Orders, y a continuación los registros dependientes de la tabla Order Details situados con un cierto nivel de indentación.

Este aspecto podemos solventarlo manipulando los campos del informe con posterioridad a su generación por el asistente, ubicándolos en los lugares que deseemos. No obstante vamos a hacer esta labor manualmente a continuación.

Tablas vinculadas manualmente

Ahora vamos a añadir otro informe al proyecto, al que daremos el nombre rptManual, realizando con el asistente sólo el paso correspondiente a la selección de las tablas Orders y Order Details. Una vez hecho esto, pulsaremos el botón Informe en blanco, pasando a la plantilla vacía del diseñador.

Seguidamente insertaremos en el informe un grupo basado en la tabla y campo Orders.OrderID, comenzando a incluir en el diseñador los campos de las tablas, objetos de texto y demás elementos hasta obtener un informe como el que muestra la Figura 56.

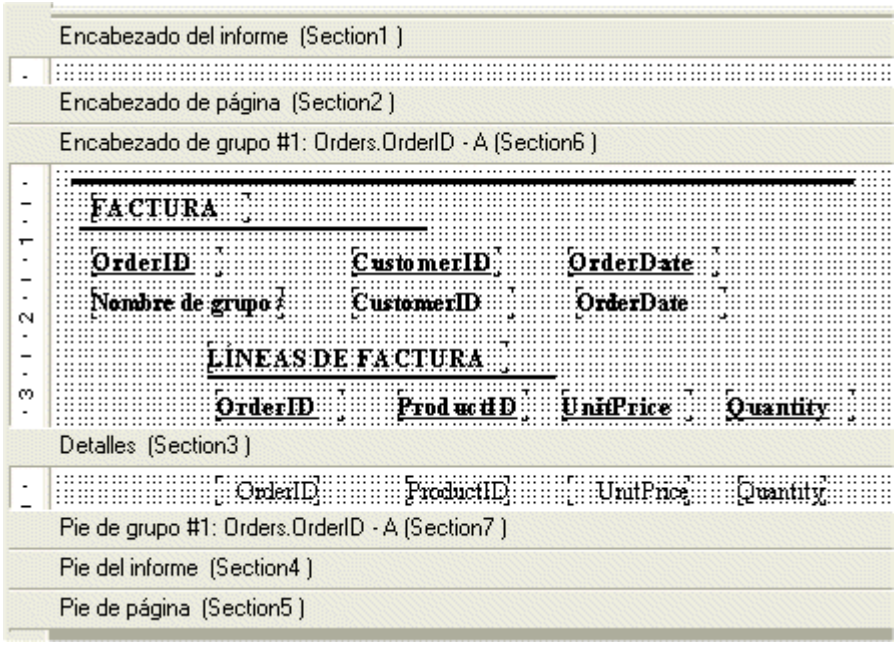


Figura 56. Informe con datos maestro-detalle creado manualmente.

Para visualizar el informe, en el evento Click( ) del otro control MenuItem del formulario escribiremos las líneas del Código fuente 8.

```
Private Sub mnuManual_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles mnuManual.Click
    Dim oRptManual As New rptManual()
    Me.crvInforme.ReportSource = oRptManual

End Sub
```

Código fuente 8

El resultado cuando seleccionemos la opción de menú *Archivo* + *Manual*, será el mostrado en la Figura 57.

En esta ocasión, hemos tenido que trabajar más en la faceta de diseño, pero la información queda distribuida más adecuadamente.

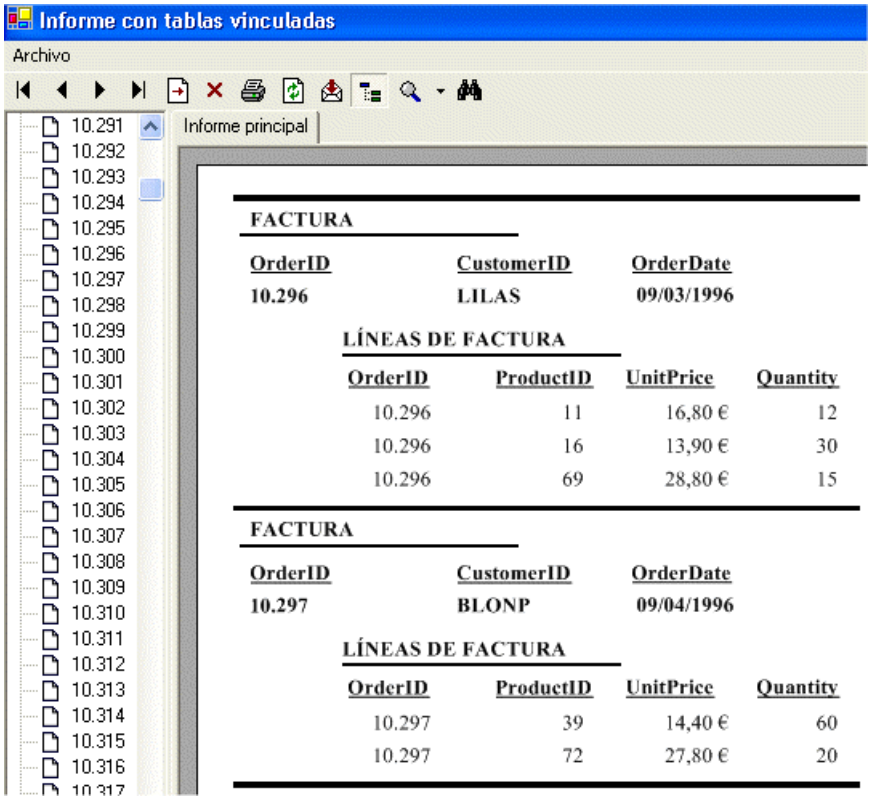


Figura 57. Informe creado manualmente, con tablas estilo maestro-detalle.

Obtención de datos relacionados entre tablas vinculadas

En los ejemplos anteriores, hemos tomado los datos directamente de las tablas sin obtener la información *cruzada* de las tablas relacionadas. Un caso concreto: cuando visualizamos los registros de las líneas de detalle, ponemos el código del producto correspondiente a la línea, pero no su nombre, esto último sería más recomendable, ya que para el usuario son más significativas las descripciones que los códigos, o en todo caso, deberíamos poner ambos.

Esta situación implica que en el informe, además de las tablas con la información elemental, también incluyamos tablas adicionales de soporte, que son las que contienen las descripciones y demás información complementaria.

A continuación vamos a desarrollar un ejemplo de listado, que partiendo de los casos de vinculación de tablas anteriores entre las tablas Orders y Order Details, mostrará también los nombres del producto en cada línea de detalle, cliente y empleado en el área de grupo, etc. Para ello crearemos un proyecto con el nombre RepVinculaRelac (para acceder al proyecto de este ejemplo hacer clic aquí).

En el asistente de creación de informes, seleccionaremos como tablas las siguientes: Customers, Employees, Products, Orders, Order Details. Los vínculos como ya hemos visto se seleccionan automáticamente, pero es posible que algún campo quede sin relacionar, por lo que será conveniente revisar todas las relaciones entre tablas y establecer las que falten. Tras seleccionar los campos que formarán parte del informe, agruparemos los registros por el campo Orders.OrderID, y finalizaremos el asistente.

Una vez que el asistente de informes haya generado su propio diseño, comenzará nuestro trabajo de modificación en la distribución de campos, personalizando el informe hasta dejarlo como muestra la Figura 58.

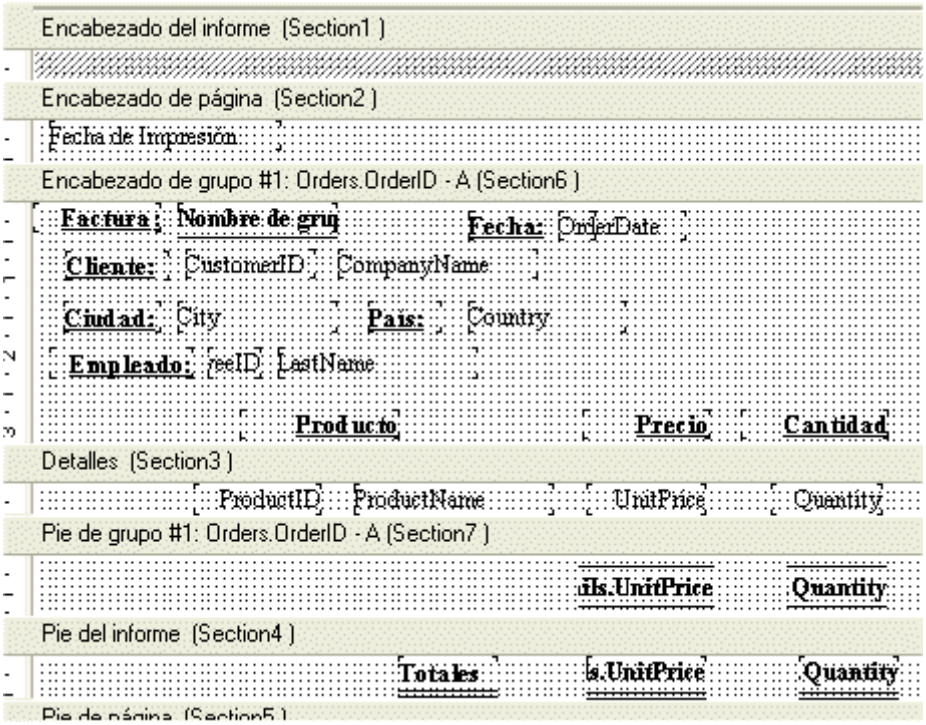


Figura 58. Diseñador de informe con tablas vinculadas mostrando campos relacionados.

Al mostrar el informe en tiempo de ejecución a través de un formulario, tendrá un aspecto como el de la Figura 59, en el que se muestra toda la información de las tablas relacionadas que hasta ahora no habíamos utilizado.

19/11/2002

Factura: 10.248 **Fecha:** 04/07/1996

Cliente: VINET Vins et alcools

Ciudad: Reims **País:** France

Empleado: 5 Buchanan

Producto		Precio	Cantidad
11	Queso Cabrales	14,00 €	12
42	Singaporean	9,80 €	10
72	Mozzarella di	34,80 €	5
		58,60 €	**27,00**

Factura: 10.249 **Fecha:** 05/07/1996

Cliente: TOMSP Toms

Ciudad: Münster **País:** Germany

Empleado: 6 Suyama

Producto		Precio	Cantidad
14	Tofu	18,60 €	9
51	Manjimup Dried	42,40 €	40
		61,00 €	**49,00**

Figura 59. Informe con campos relacionados de múltiples tablas.

Fórmulas

Una fórmula es un fragmento de código que situamos en forma de control o campo dentro del informe, y que es evaluada por el motor de Crystal Reports cuando el listado se ejecuta.

El número de veces que una fórmula será evaluada en un informe depende de la sección del mismo en el que sea situada. Así pues, una fórmula que se encuentre en la sección Detalles será ejecutada por cada registro que se muestre en el informe, mientras que una ubicada en el pie del informe sólo se ejecutará en una ocasión.

Las fórmulas son útiles, para situaciones en las que no exista un campo en la tabla que proporcione determinada información, o bien el propio Crystal Reports no disponga de un objeto que lo suministre.

Como ejemplo práctico del uso de fórmulas, crearemos un proyecto con el nombre RepFormulas (para acceder al proyecto de este ejemplo hacer clic aquí), al que añadiremos un informe que deberá visualizar la tabla Order Details.

Esta tabla muestra la información de las líneas de detalle de cada factura emitida en la base de datos Northwind. Entre otros datos, tenemos el precio unitario y la cantidad del producto vendida para una línea de factura; sin embargo, no disponemos del importe para cada línea, que obtendríamos al multiplicar el precio por la cantidad.

Bien es cierto que este valor podemos obtenerlo creando una consulta con un campo calculado que nos devuelva el dato, pero en este caso vamos a conseguirlo desde el informe, a través de la creación de una fórmula.

Para crear una fórmula nos situaremos en el Explorador de campos, haciendo clic derecho en el nodo *Campos de fórmula*, y seleccionando la opción *Nuevo* del menú contextual.

Tras introducir el nombre de la fórmula, en este caso Importe, se abrirá la ventana *Editor de fórmulas*, en la que podemos distinguir varios elementos de trabajo: editor de código, campos de la tabla, funciones, etc. Ver la Figura 60.

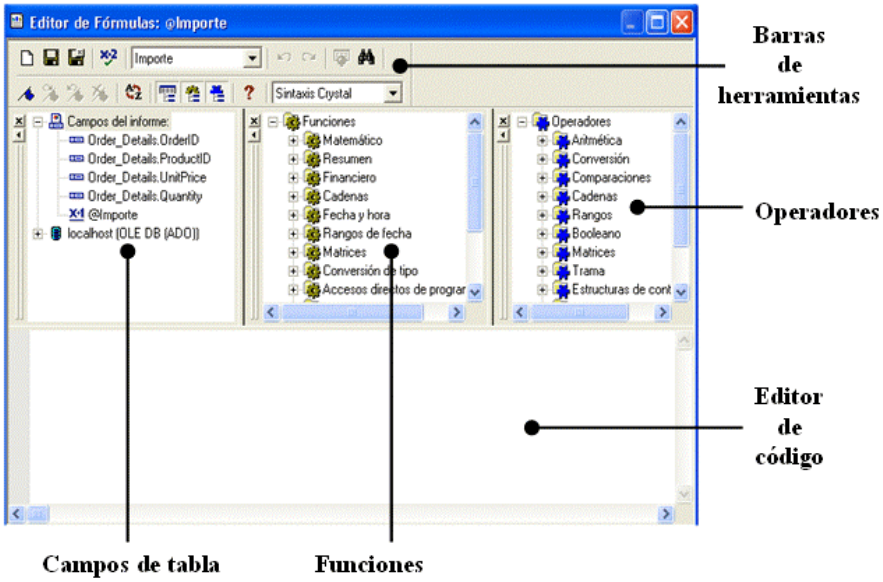


Figura 60. Ventana del editor de fórmulas.

Para escribir el código de la fórmula podemos emplear dos tipos de sintaxis: Crystal o Basic; en este caso utilizaremos la primera, ya que es el tipo de sintaxis que propone por defecto este editor. Consulte el lector, la documentación correspondiente a Crystal Reports en Visual Studio .NET para mayor información sobre las características de la sintaxis para fórmulas.

El modo de seleccionar los diferentes elementos de esta ventana correspondientes a campos, funciones y operadores, consiste en hacer doble clic en cualquiera de ellos, trasladándose de esta manera, al editor de código.

En primer lugar, seleccionaremos del apartado de campos del informe, el campo UnitPrice. A continuación, en el apartado Operadores, expandiremos el nodo Aritmética, seleccionando el operador Multiplicar. Finalmente añadiremos el campo Quantity, quedando la fórmula como vemos en la Figura 61.

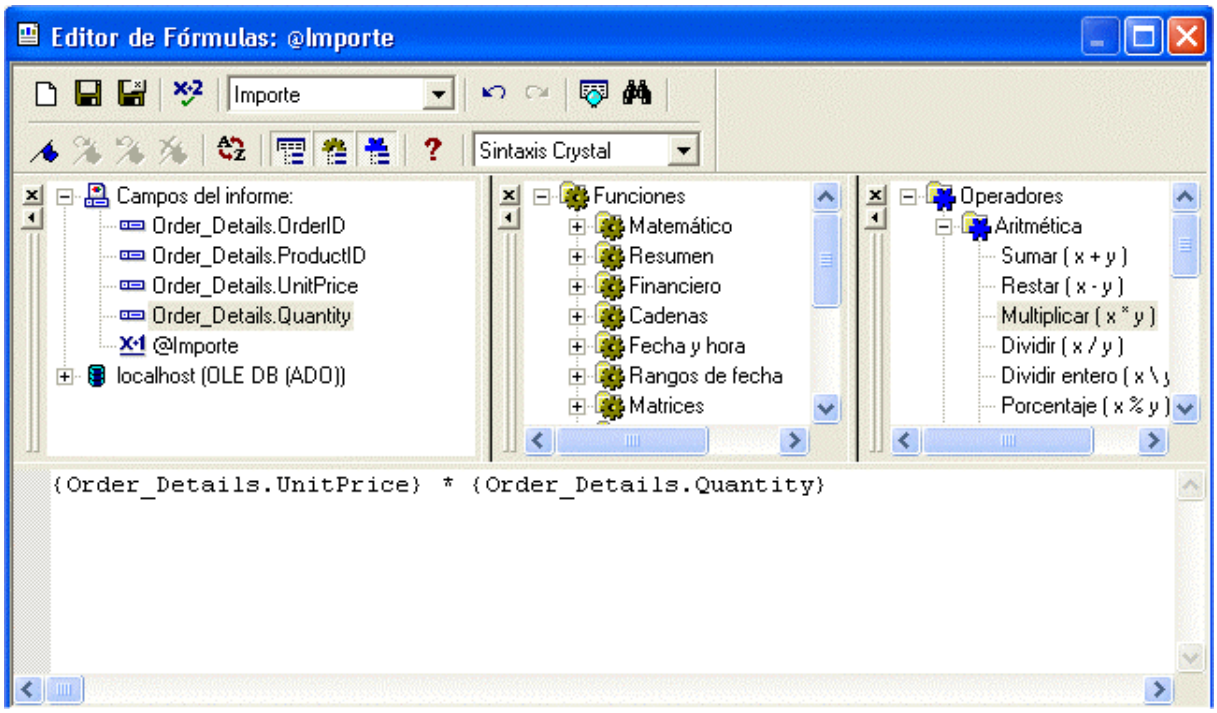


Figura 61. Resultado del código de la fórmula.

Terminaremos pulsando en la primera barra de herramientas de esta ventana, el botón para guardar la fórmula, que corresponde al tercero comenzando por la izquierda, de manera que nuestra fórmula quedará agregada al nodo Campos de fórmula, de la ventana Explorador de campos.

Para utilizar la fórmula en nuestro informe, haremos clic en ella y la arrastraremos hasta la sección Detalles, al igual que hacemos con un campo normal de la tabla.

Completaremos un poco más este listado creando una nueva fórmula con el nombre TipoPedido, que en función del valor del campo Quantity, devuelva una cadena de caracteres indicando si el pedido de producto para cada línea es pequeño, mediano o grande. El Código fuente 9 muestra el código que necesitaremos incluir para esta fórmula.

```
select {Order_Details.Quantity}
case 1 To 40 :
    "PEQUEÑO"
```

```
case 41 To 70 :
    "MEDIANO"
case Is > 71 :
    "GRANDE"
```

Código fuente 9

El resultado final con el informe incluyendo ambas fórmulas en los detalles del listado, se muestra en la Figura 62.

OrderID	ProductID	UnitPrice	Quantity	Importe	TipoPedido
10.318	76	14,40 €	6	86,40 €	PEQUEÑO
10.319	17	31,20 €	8	249,60 €	PEQUEÑO
10.319	28	36,40 €	14	509,60 €	PEQUEÑO
10.319	76	14,40 €	30	432,00 €	PEQUEÑO
10.320	71	17,20 €	30	516,00 €	PEQUEÑO
10.321	35	14,40 €	10	144,00 €	PEQUEÑO
10.322	52	5,60 €	20	112,00 €	PEQUEÑO
10.323	15	12,40 €	5	62,00 €	PEQUEÑO
10.323	25	11,20 €	4	44,80 €	PEQUEÑO
10.323	39	14,40 €	4	57,60 €	PEQUEÑO
10.324	16	13,90 €	21	291,90 €	PEQUEÑO
10.324	35	14,40 €	70	1.008,00 €	MEDIANO
10.324	46	9,60 €	30	288,00 €	PEQUEÑO
10.324	59	44,00 €	40	1.760,00 €	PEQUEÑO
10.324	63	35,10 €	80	2.808,00 €	GRANDE
10.325	6	20,00 €	6	120,00 €	PEQUEÑO
10.325	13	4,80 €	12	57,60 €	PEQUEÑO
10.325	14	18,60 €	9	167,40 €	PEQUEÑO

Figura 62. Informe incluyendo los campos de fórmula Importe y TipoPedido.

Suprimir valores duplicados

Al ejecutar el listado del ejemplo anterior comprobamos que la fórmula TipoPedido repite en muchas ocasiones su valor, efecto que en algunos casos puede ser molesto.

Para evitar esta circunstancia, haremos clic derecho sobre este campo de fórmula, seleccionando la opción *Formato* del menú contextual, y en la ventana del editor de formato, marcaremos el CheckBox Suprimir si se duplica. Ver la Figura 63.

Si ahora volvemos a ejecutar este informe, su aspecto variará evitando mostrar valores duplicados en el campo de fórmula TipoPedido, como vemos en la Figura 64.

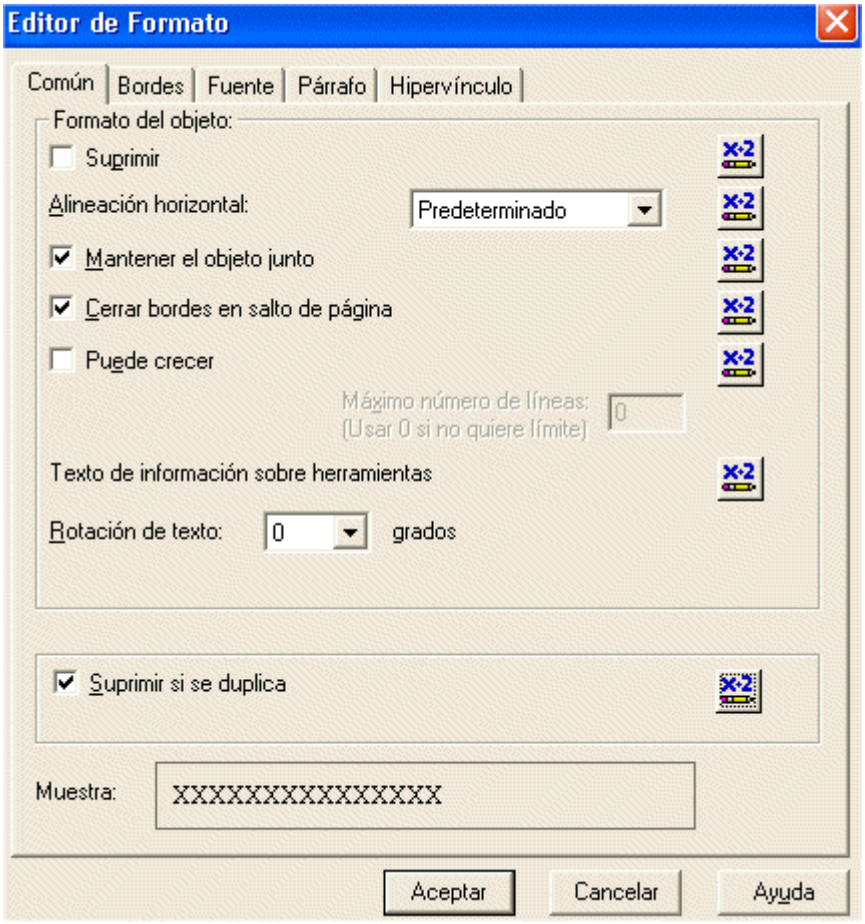


Figura 63. Configurar campo de fórmula para no repetir valores en el informe.

OrderID	ProductID	UnitPrice	Quantity	Importe	TipoPedido
10.318	76	14,40 €	6	86,40 €	PEQUEÑO
10.319	17	31,20 €	8	249,60 €	
10.319	28	36,40 €	14	509,60 €	
10.319	76	14,40 €	30	432,00 €	
10.320	71	17,20 €	30	516,00 €	
10.321	35	14,40 €	10	144,00 €	
10.322	52	5,60 €	20	112,00 €	
10.323	15	12,40 €	5	62,00 €	
10.323	25	11,20 €	4	44,80 €	
10.323	39	14,40 €	4	57,60 €	
10.324	16	13,90 €	21	291,90 €	
10.324	35	14,40 €	70	1.008,00 €	MEDIANO
10.324	46	9,60 €	30	288,00 €	PEQUEÑO
10.324	59	44,00 €	40	1.760,00 €	
10.324	63	35,10 €	80	2.808,00 €	GRANDE
10.325	6	20,00 €	6	120,00 €	PEQUEÑO
10.325	13	4,80 €	12	57,60 €	
10.325	14	18,60 €	9	167,40 €	
10.325	31	10,00 €	4	40,00 €	

Figura 64. Informe sin duplicados en la fórmula TipoPedido.

Modificación de la fórmula de selección

Cuando a través del asistente para crear informes, añadimos una selección o filtro de datos para mostrar un subconjunto de registros de la tabla, dicha selección queda guardada en el informe como una fórmula, que podremos modificar con posterioridad si es necesario.

En el presente apartado vamos a crear un proyecto con el nombre RepFormulaSel (para acceder al proyecto de este ejemplo hacer clic aquí), al que añadiremos un informe que obtendrá datos de la tabla Order Details.

En el paso correspondiente a la selección de datos del asistente, estableceremos el campo OrderID junto a un rango de selección, como vemos en la Figura 65.

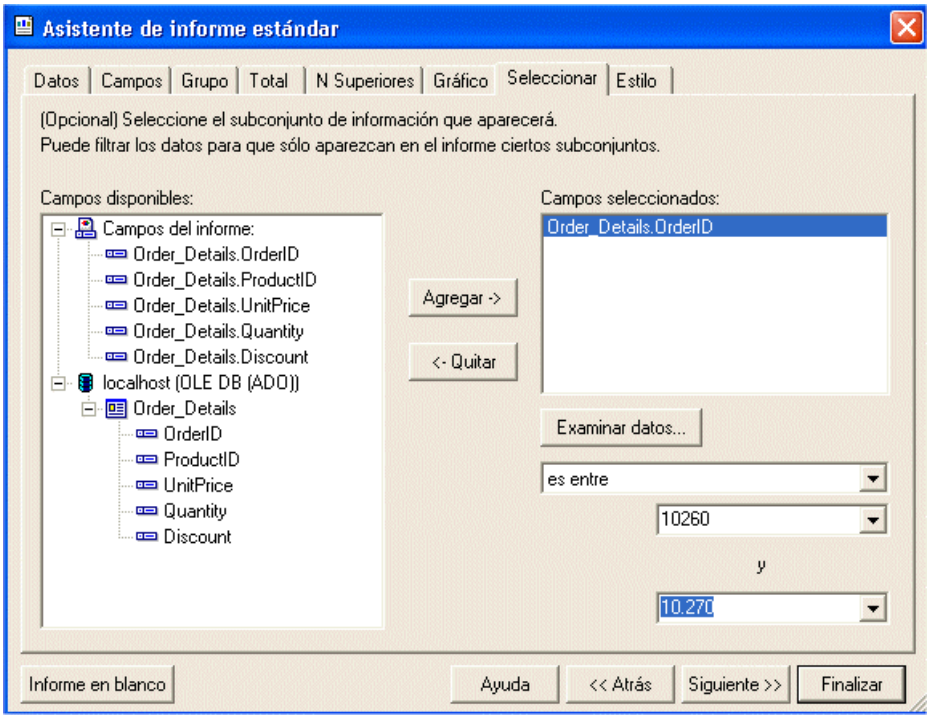


Figura 65. Selección de intervalo de registros para el informe.

Tras añadir el código para mostrar el informe en el evento Load( ) del formulario, podremos comprobar que efectivamente, los registros de la tabla se encuentran en el intervalo indicado. Ver el Código fuente 10.

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    Dim oRptPrueba As New rptPrueba()
    Me.crvInforme.ReportSource = oRptPrueba
End Sub
```

Código fuente 10

A continuación vamos a modificar la fórmula de selección de registros haciendo clic derecho en el diseñador del informe, y seleccionando en el menú contextual *Informe + Editar fórmula de selección + Registros*.

Esta acción abrirá la ventana del editor de fórmulas, en la que introduciremos el cambio de intervalo como muestra la Figura 66.

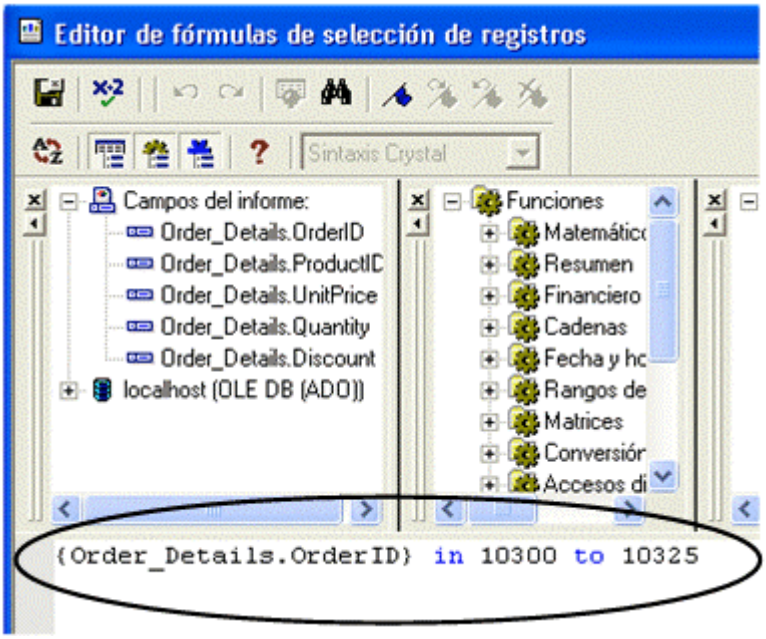


Figura 66. Fórmula de selección de registros.

Al volver a ejecutar el informe, comprobaremos que los registros en el listado están entre el nuevo rango. Ver la Figura 67.

OrderID	ProductID	UnitPrice
10.300	66	13,60 €
10.300	68	10,00 €
10.301	40	14,70 €
10.301	56	30,40 €
10.302	17	31,20 €
10.302	28	36,40 €
10.302	43	36,80 €
10.303	40	14,70 €
10.303	65	16,80 €
10.303	68	10,00 €
10.304	49	16,00 €
10.304	59	44,00 €
10.304	71	17,20 €

Figura 67. Informe con nueva selección de registros.

Aplicar formato a una sección

En función del resultado devuelto por una expresión podemos aplicar un formato de visualización a las filas de detalle de un informe, para destacar determinados registros.

Crearemos para este ejemplo un proyecto con el nombre RepFormatSeccion (para acceder al proyecto de este ejemplo hacer clic aquí) y añadiremos un informe que muestre de forma vinculada, las tablas Orders y Order Details.

A continuación haremos clic derecho en el título de la sección Detalles, seleccionando la opción *Dar formato a la sección* del menú contextual, que abrirá la ventana del asistente de la sección. Ver la Figura 68.

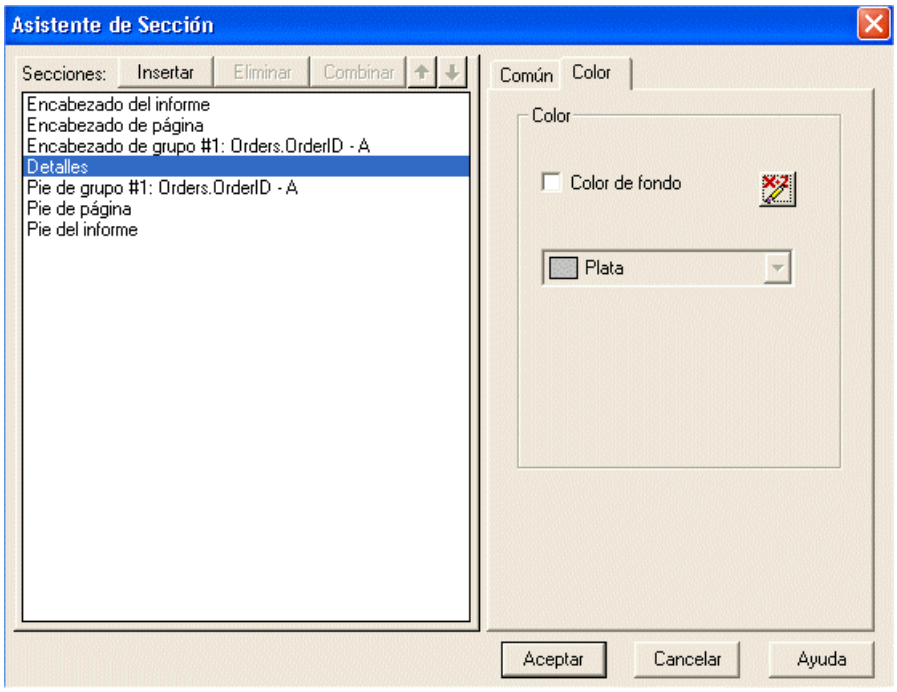


Figura 68. Asistente de sección.

Después haremos clic en la pestaña Color, y pulsaremos el botón de fórmula, situado junto al CheckBox Color de fondo. Al abrirse el editor de fórmulas escribiremos el código mostrado en la Figura 69.

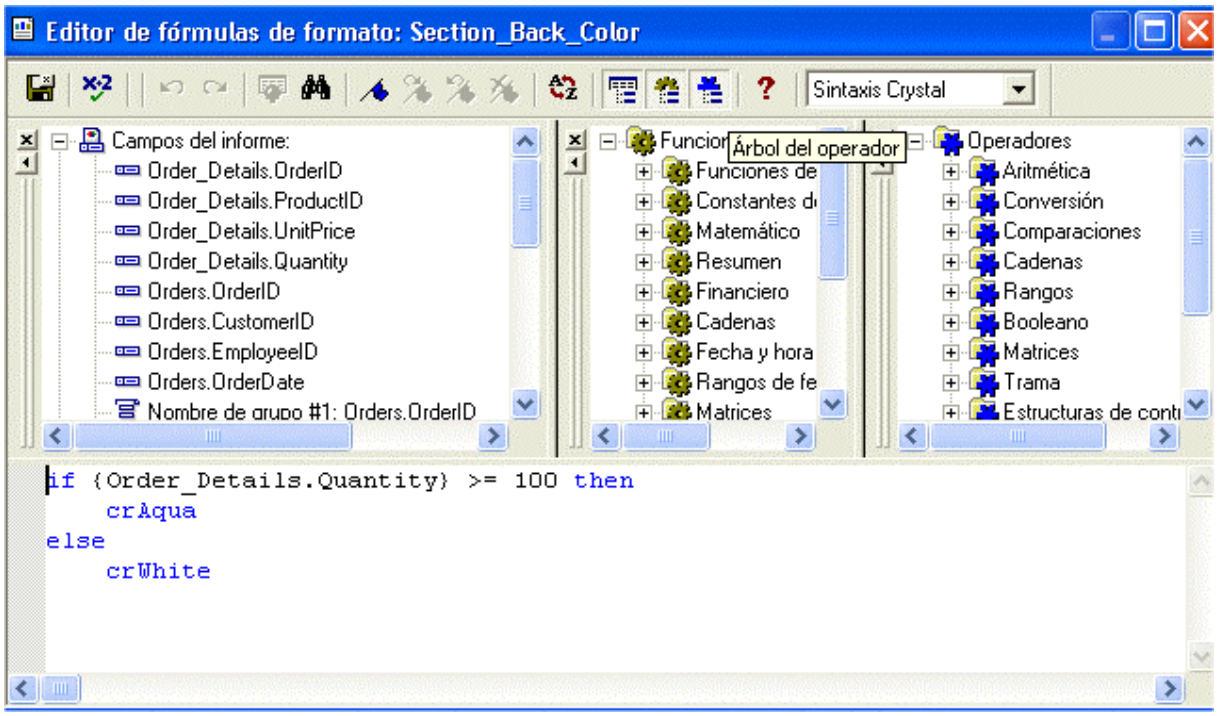
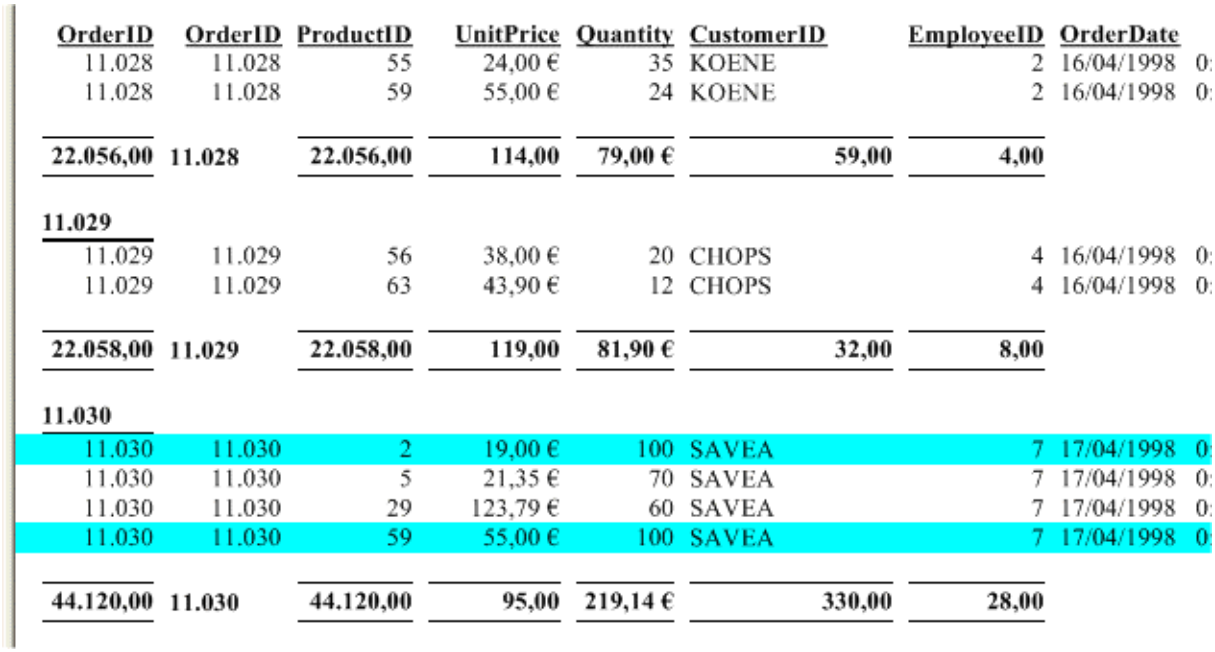


Figura 69. Fórmula de selección para el color de fondo de una sección.

La operación que realizamos a través de esta fórmula es la siguiente: cuando el campo Quantity del registro a mostrar es mayor o igual que 100, cambiamos el color de fondo de la línea a aguamarina, y en caso contrario ponemos el color blanco (utilizamos las constantes de color crAqua y crWhite). La Figura 70 muestra la ejecución del informe con registros que cumplen la condición para el cambio de color.



OrderID	OrderID	ProductID	UnitPrice	Quantity	CustomerID	EmployeeID	OrderDate	
11.028	11.028	55	24,00 €	35	KOENE	2	16/04/1998	0:
11.028	11.028	59	55,00 €	24	KOENE	2	16/04/1998	0:
22.056,00	**11.028**	**22.056,00**	**114,00**	**79,00 €**	**59,00**	**4,00**		
11.029								
11.029	11.029	56	38,00 €	20	CHOPS	4	16/04/1998	0:
11.029	11.029	63	43,90 €	12	CHOPS	4	16/04/1998	0:
22.058,00	**11.029**	**22.058,00**	**119,00**	**81,90 €**	**32,00**	**8,00**		
11.030								
11.030	11.030	2	19,00 €	100	SAVEA	7	17/04/1998	0:
11.030	11.030	5	21,35 €	70	SAVEA	7	17/04/1998	0:
11.030	11.030	29	123,79 €	60	SAVEA	7	17/04/1998	0:
11.030	11.030	59	55,00 €	100	SAVEA	7	17/04/1998	0:
44.120,00	**11.030**	**44.120,00**	**95,00**	**219,14 €**	**330,00**	**28,00**		

Figura 70. Informe con registros cambiados de color a través de la fórmula de la sección.

Creación de totales y subtotales

En el caso de que tengamos que crear informes con una o varias columnas que muestren valores numéricos (por ejemplo importes), con toda seguridad será necesaria la suma de dichos datos para obtener un total general al final del listado, y en el caso de que el informe se encuentre agrupado por un determinado campo, posiblemente también será conveniente calcular un subtotal por grupo.

A lo largo de este apartado explicaremos el modo de creación de totales y subtotales, utilizando tanto el asistente de creación de informes como el modo manual. Para ello crearemos un proyecto de ejemplo con el nombre RepTotal (para acceder al proyecto de este ejemplo hacer clic aquí).

Uso del asistente de informes para crear totales y subtotales

En el primer informe que vamos a diseñar, y al que daremos el nombre rptAsistente, utilizaremos el asistente para la creación de todos los totales. La tabla de la base de datos a utilizar será Order Details.

Completaremos todos los pasos del asistente de informes como ya hemos hecho en anteriores ocasiones, teniendo presente que debemos crear un grupo por el campo OrderID.

Al llegar a la pestaña Total el asistente revisa los campos de tipo numérico de la tabla, y automáticamente los sitúa en el panel *Campos resumidos*; en nuestro caso sólo vamos a dejar en este panel el campo UnitPrice. Para crear un total general por los campos de resumen debemos marcar la casilla Agregar totales generales en esta etapa del asistente. Observe también el lector que por defecto

la operación de resumen es una suma, pero es posible cambiarla en la lista desplegable Tipo de resumen, por una operación de promedio, recuento, etc. Ver la Figura 71.

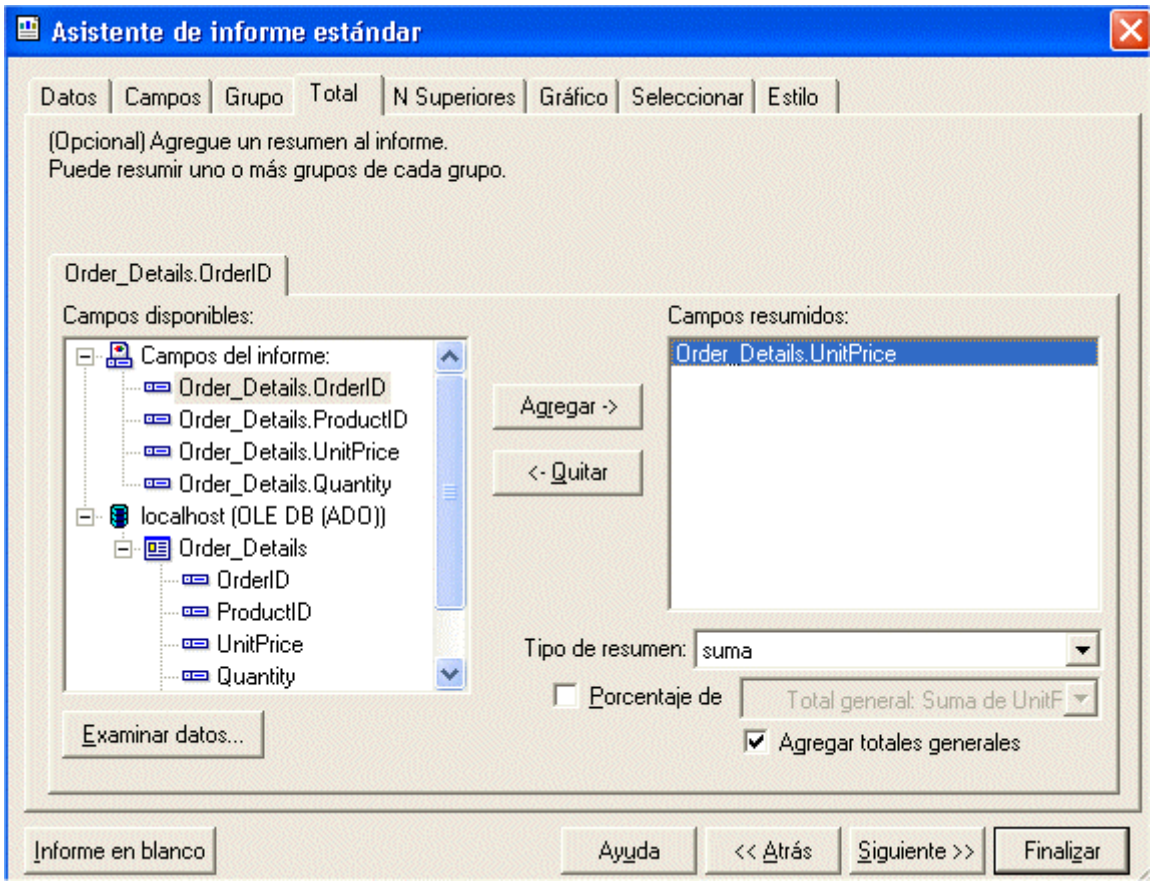


Figura 71. Creación de totales y subtotales con el asistente.

Seguiremos con el asistente hasta el paso Seleccionar, en el que incluiremos un rango de valores para el campo OrderID, de modo que el listado no genere un número elevado de páginas. En este punto finalizaremos el asistente.

Para visualizar este informe en un control CrystalReportViewer del formulario del proyecto, insertaremos un menú al formulario, y le añadiremos una opción con el nombre Asistente, de modo que al seleccionarla el usuario, se muestre el listado de la Figura 72.

Podemos comprobar en la figura anterior que para cada grupo se realiza una suma independiente por el campo UnitPrice sólo de los registros del grupo. Finalmente se realiza una suma general de este mismo campo, cuyo resultado se muestra tras el último registro.

14/11/2002

OrderID	ProductID	UnitPrice	Quantity
10.275			
10.275	24	3,60 €	12
10.275	59	44,00 €	6
10.275		**47,60 €**	
10.276			
10.276	10	24,80 €	15
10.276	13	4,80 €	10
10.276		**29,60 €**	
Total general:		**1.438,30 €**	

Figura 72. Listado con totales y subtotales generado con el asistente.

Creación manual de un total general

Para esta ocasión añadiremos al proyecto un informe con el nombre rptManualTotal, que mostrará el contenido de la tabla Products. Tras establecer la tabla de trabajo en el paso Datos del asistente, lo finalizaremos pulsando el botón Informe en blanco.

Añadiremos manualmente algunos campos de la tabla, siendo importante que incluyamos el campo UnitPrice, puesto que este será por el que generemos el total.

A continuación haremos clic derecho en el diseñador de informes y seleccionaremos la opción del menú contextual *Insertar + Total general*, que abrirá la ventana para realizar esta operación, en la que seleccionaremos el campo a totalizar como vemos en la Figura 73.

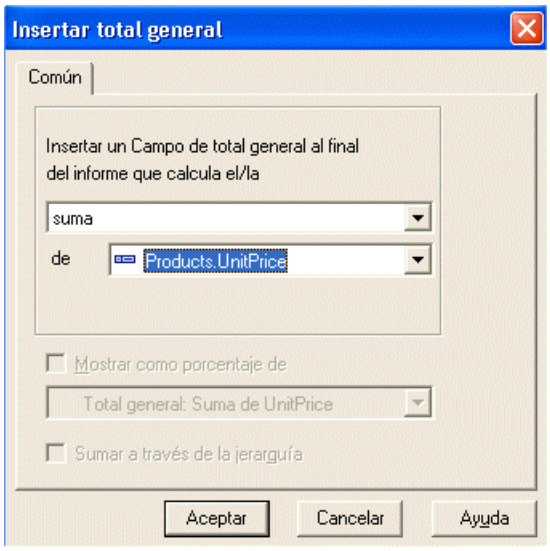


Figura 73. Inserción manual de total general para el informe.

Como resultado se añadirá en la sección Pie del informe el nuevo campo correspondiente al total creado. Ver Figura 74.

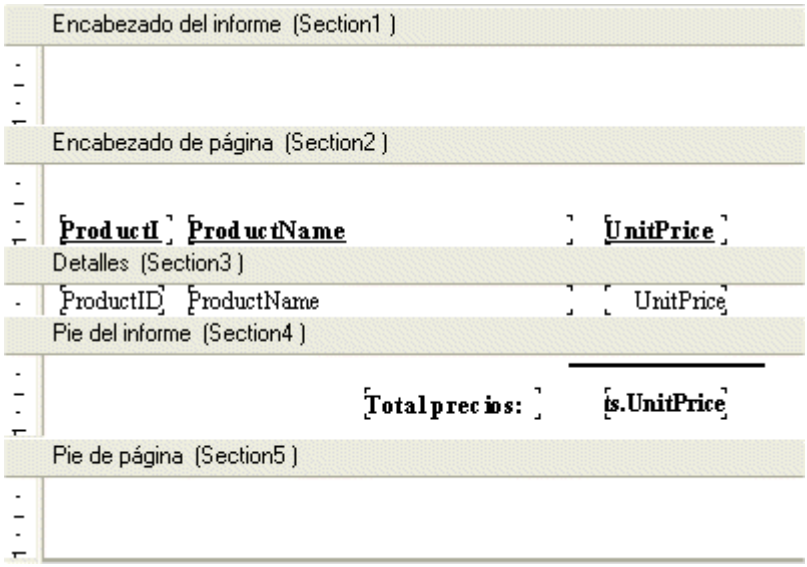


Figura 74. Diseñador de informe con campo de total general.

Agregaremos al menú del formulario una opción con el nombre *Manual Total general*, en la que escribiremos el código que carga este informe en el control visualizador. La Figura 75 muestra el informe en ejecución.

ProductI	ProductName	UnitPrice
61	Sirop d'érable	28,50 €
62	Tarte au sucre	49,30 €
63	Vegie-spread	43,90 €
64	Wimmers gute Semmelknödel	33,25 €
65	Louisiana Fiery Hot Pepper Sauce	21,05 €
66	Louisiana Hot Spiced Okra	17,00 €
67	Laughing Lumberjack Lager	14,00 €
68	Scottish Longbreads	12,50 €
69	Gudbrandsdalsost	36,00 €
70	Outback Lager	15,00 €
71	Flotemysost	21,50 €
72	Mozzarella di Giovanni	34,80 €
73	Röd Kaviar	15,00 €
74	Longlife Tofu	10,00 €
75	Rhönbräu Klosterbier	7,75 €
76	Lakkalikööri	18,00 €
77	Original Frankfurter grüne Soße	13,00 €
	Total precios:	**2.222,71 €**

Figura 75. Informe con total general.

Creación manual de un subtotal

De igual modo que en el apartado anterior, añadiremos ahora un informe al proyecto con el nombre rptManualSubtotal, que mostrando también los registros de la tabla Products, en esta ocasión incluirá un campo de subtotal.

Una vez que en el asistente hemos establecido cuál será la tabla a utilizar, finalizaremos el asistente creando un informe en blanco, y añadiremos los mismos campos que en el informe anterior.

Seguidamente haremos clic derecho en el informe, seleccionando la opción de menú contextual *Insertar* + *Subtotal*, que mostrará la ventana para realizar esta operación, en la que seleccionaremos el campo UnitPrice para crear el subtotal, y el campo CategoryID para realizar la agrupación de registros. Ver la Figura 76.

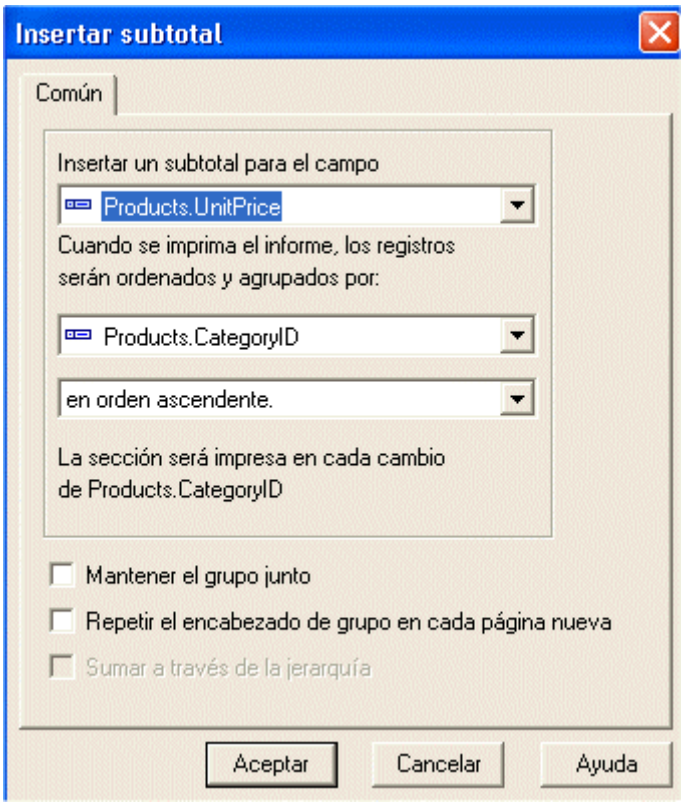


Figura 76. Ventana para la creación manual de un subtotal.

Al pulsar Aceptar se creará en el diseñador del informe un nuevo grupo, cuyo pie incluirá el campo de subtotal, como vemos en la Figura 77.

Para poder usar este informe en ejecución, añadiremos al menú del formulario una opción con el nombre *Manual Subtotal*, que al ejecutarla mostrará el informe como vemos en la Figura 78.

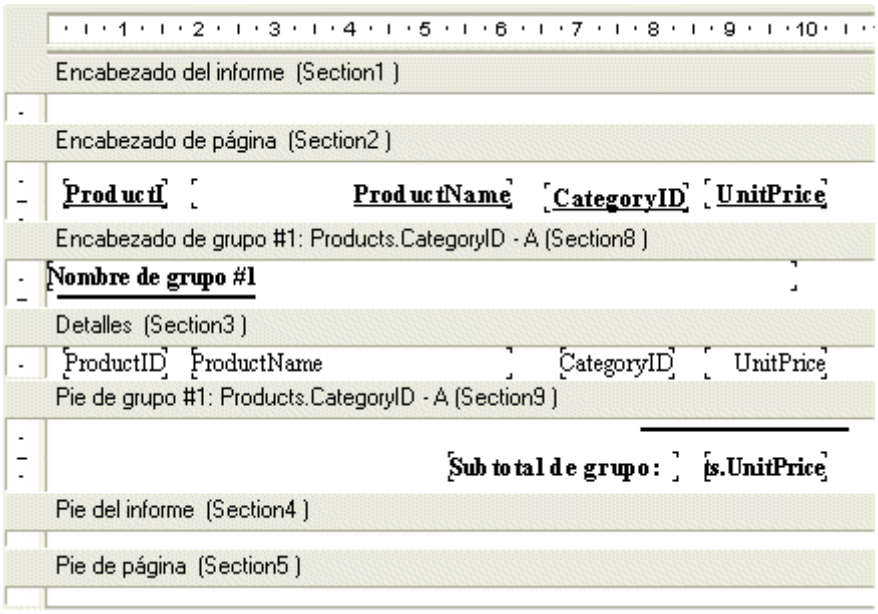
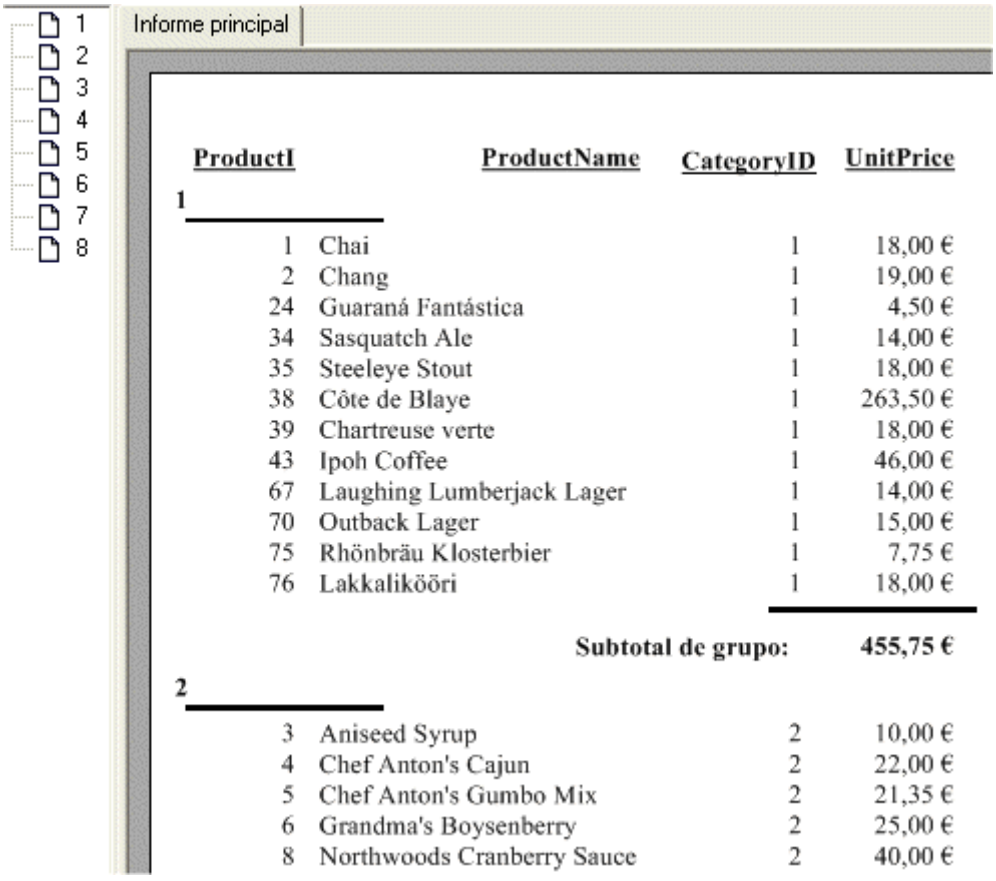


Figura 77. Informe con campo de subtotal.



ProductI	ProductName	CategoryID	UnitPrice
1			
1	Chai	1	18,00 €
2	Chang	1	19,00 €
24	Guaraná Fantástica	1	4,50 €
34	Sasquatch Ale	1	14,00 €
35	Steeleye Stout	1	18,00 €
38	Côte de Blaye	1	263,50 €
39	Chartreuse verte	1	18,00 €
43	Ipoh Coffee	1	46,00 €
67	Laughing Lumberjack Lager	1	14,00 €
70	Outback Lager	1	15,00 €
75	Rhönbräu Klosterbier	1	7,75 €
76	Lakkalikööri	1	18,00 €
	Subtotal de grupo:		**455,75 €**
2			
3	Aniseed Syrup	2	10,00 €
4	Chef Anton's Cajun	2	22,00 €
5	Chef Anton's Gumbo Mix	2	21,35 €
6	Grandma's Boysenberry	2	25,00 €
8	Northwoods Cranberry Sauce	2	40,00 €

Figura 78. Informe con subtotales.

Inclusión de un gráfico estadístico en el informe

Cuando creamos un informe, hay ocasiones en las que mostrar los datos en modo textual no es suficiente para que el usuario del programa obtenga la información que necesita.

Por ejemplo, en el caso de informes que necesitan proporcionar una estadística de resultados, es muy recomendable acompañar junto a los datos tradicionales, una gráfica ilustrativa. En este tipo de escenarios es donde la premisa: *una imagen vale más que mil palabras*, alcanza su mayor significado.

El potente motor de generación de informes de CR, entre sus muchas funcionalidades, nos aporta también la capacidad de generar gráficos estadísticos. Como ejemplo ilustrativo, vamos a crear un proyecto con el nombre RepGrafico (para acceder al proyecto de este ejemplo hacer clic aquí), añadiendo un informe que configuraremos en su asistente para que obtenga los datos de las tablas Employees y Orders, y en el que crearemos también un grupo por el campo Orders.EmployeeID. Como no es el objetivo de este informe la obtención de totales, quitaremos en el paso Total, los campos de total automáticos creados por el asistente.

Al llegar en el asistente a la pestaña Gráfico, disponemos de multitud de tipos distintos de gráficos para añadir, y estilos para configurarlo; en nuestro caso diseñaremos un gráfico que nos proporcione un estudio de las ventas realizadas a clientes por cada empleado.

Como tipo de gráfico elegiremos Circular, en cuanto al estilo, que estableceremos a la derecha la ventana actual, dejaremos el que viene por defecto. Ver la Figura 79.

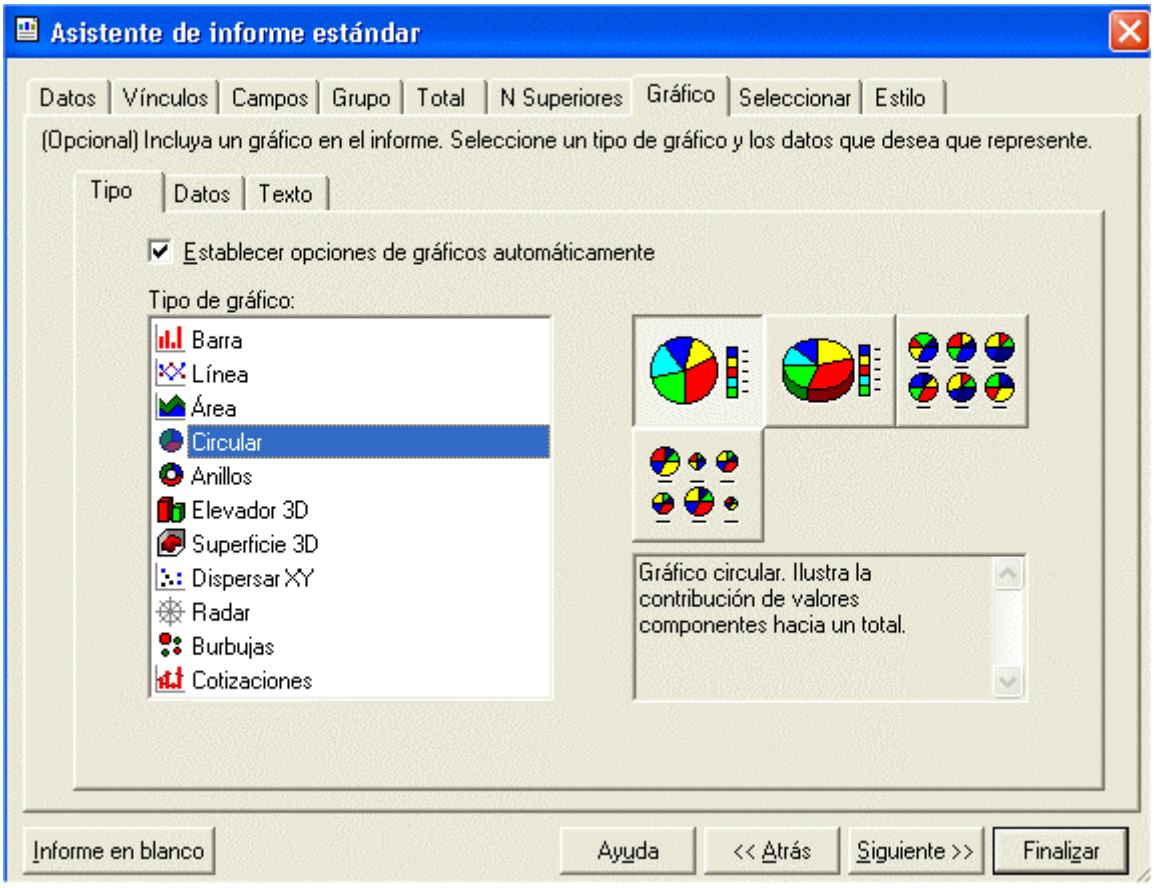


Figura 79. Elección del tipo y estilo de gráfico.

Pulsando en la pestaña Datos, en el paso actual del asistente, estableceremos la ubicación del gráfico en el informe y el comportamiento a la hora de recolectar los datos para componer el gráfico. En este sentido, lo que haremos será, que por cada empleado se sumarán las facturas que ha emitido, componiendo una porción del círculo. Ver la Figura 80.

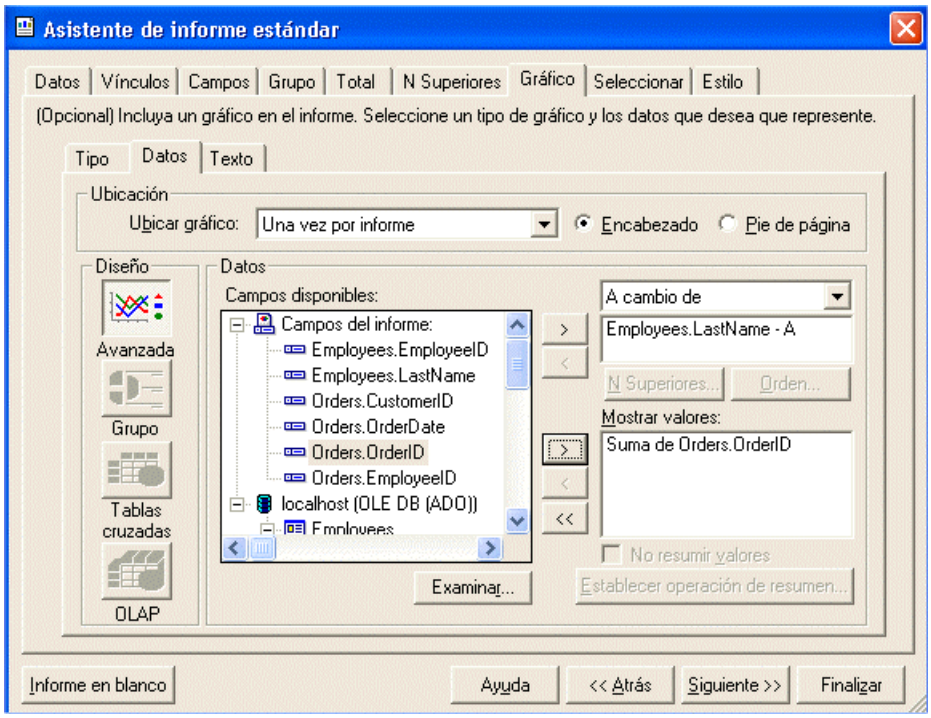


Figura 80. Selección del modo de creación del gráfico según sus datos.

En la pestaña Texto podemos escribir una cadena para el título, subtítulo y la nota al pie del gráfico. Por defecto, estos valores son automáticos, de modo que para escribir nuestro propio texto tenemos que desmarcar la casilla que necesitemos, como muestra la Figura 81.

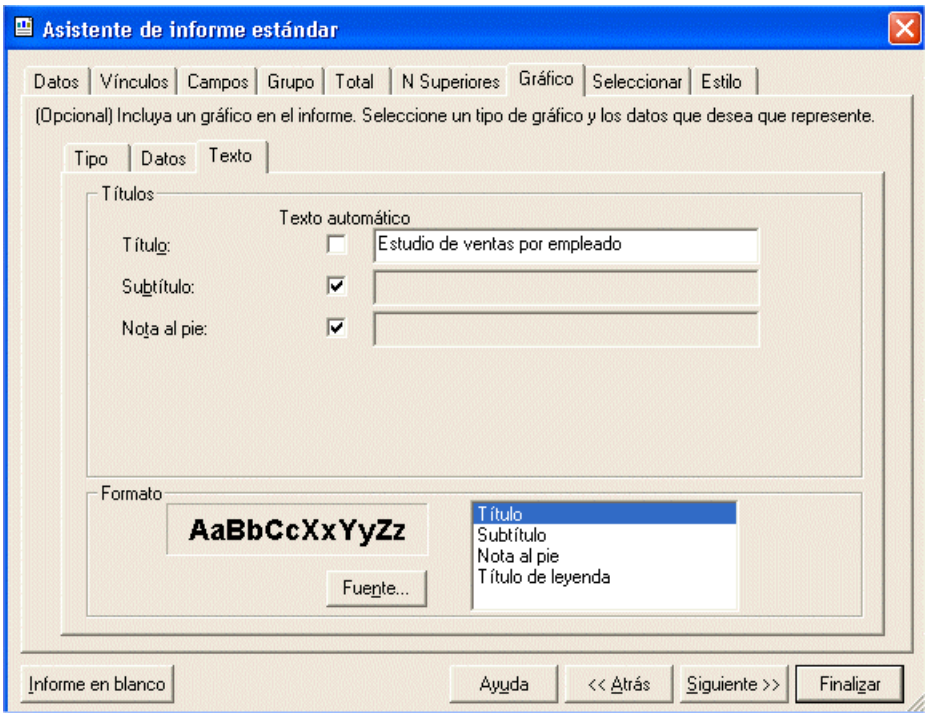


Figura 81. Campos de texto para adjuntar al gráfico.

Tras dar los anteriores pasos, ya habríamos terminado de confeccionar el gráfico y por ende, el informe. Pulsaremos Finalizar en el asistente, generándose el informe, con la novedad en esta ocasión de que el gráfico que hemos creado se incluirá en la sección Encabezado del informe. Tenga en cuenta el lector, que el gráfico que se muestra en modo diseño es meramente una referencia de posición y tamaño dentro del informe, y no proporciona valores reales, puesto que todavía no ha realizado una conexión al origen de datos para extraer la información, cosa que sucederá al ejecutarlo. Ver Figura 82.

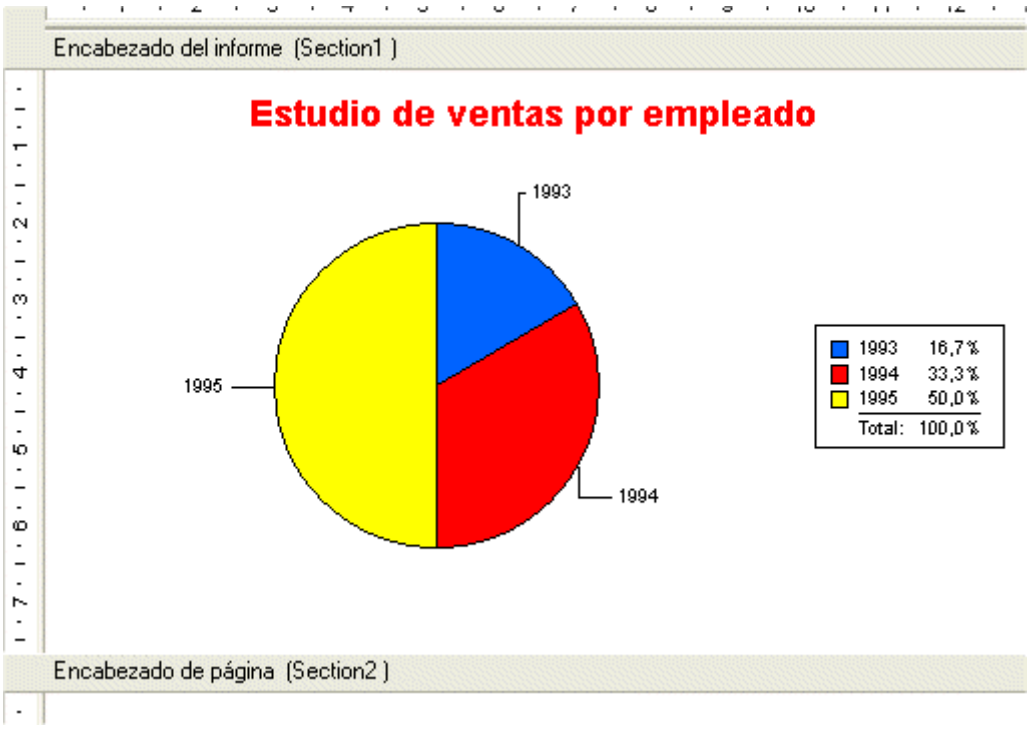


Figura 82. Gráfico en el diseñador del informe.

En cuanto a los campos del informe, realizaremos algunos cambios en su distribución para mostrarlos en un modo personalizado, con una organización distinta a la generada por el asistente. Ver Figura 83.

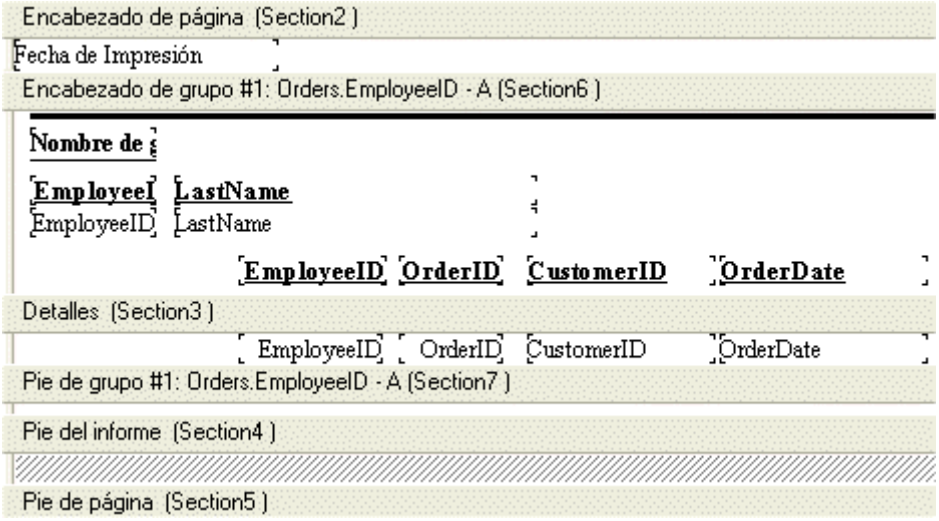


Figura 83. Distribución de campos en el informe con gráfico.

La Figura 84 muestra el resultado del informe en tiempo de ejecución.

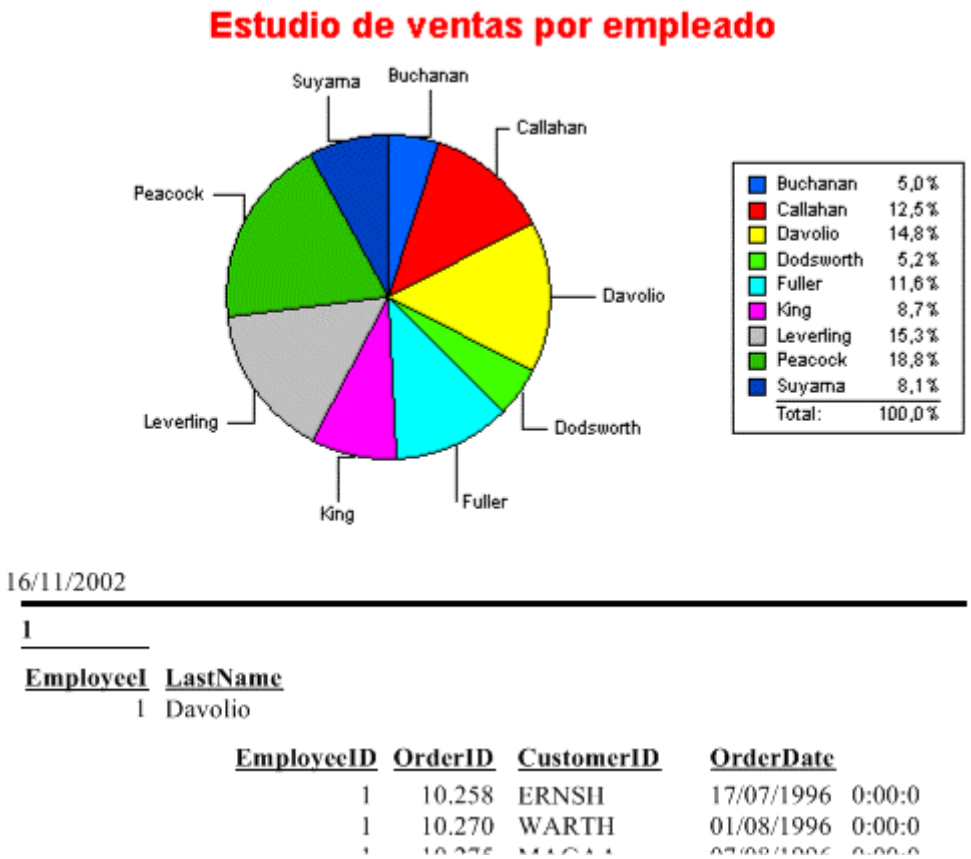


Figura 84. Ejecución del informe con gráfico estadístico.

Manipulación por código de los elementos del informe

Otra de las facetas destacables en Crystal Reports es su capacidad de manipular diversos aspectos de un informe desde código. Si bien, esta cualidad ya estaba presente en anteriores versiones de CR, se trata de un aspecto que se ha potenciado todavía más en la actual versión de este generador de informes, permitiéndonos por ejemplo, que un mismo informe visualice conjuntos diferentes de registros en distintas ejecuciones, a través de parámetros pasados por código.

En los siguientes apartados se tratarán alguno de los principales aspectos en cuanto a la manipulación de un informe en tiempo de ejecución por código.

Título del informe

Cuando creamos un informe desde su asistente, en el último paso del mismo, que recibe el nombre Estilo, podemos asignar un título para el listado como vemos en la Figura 85.

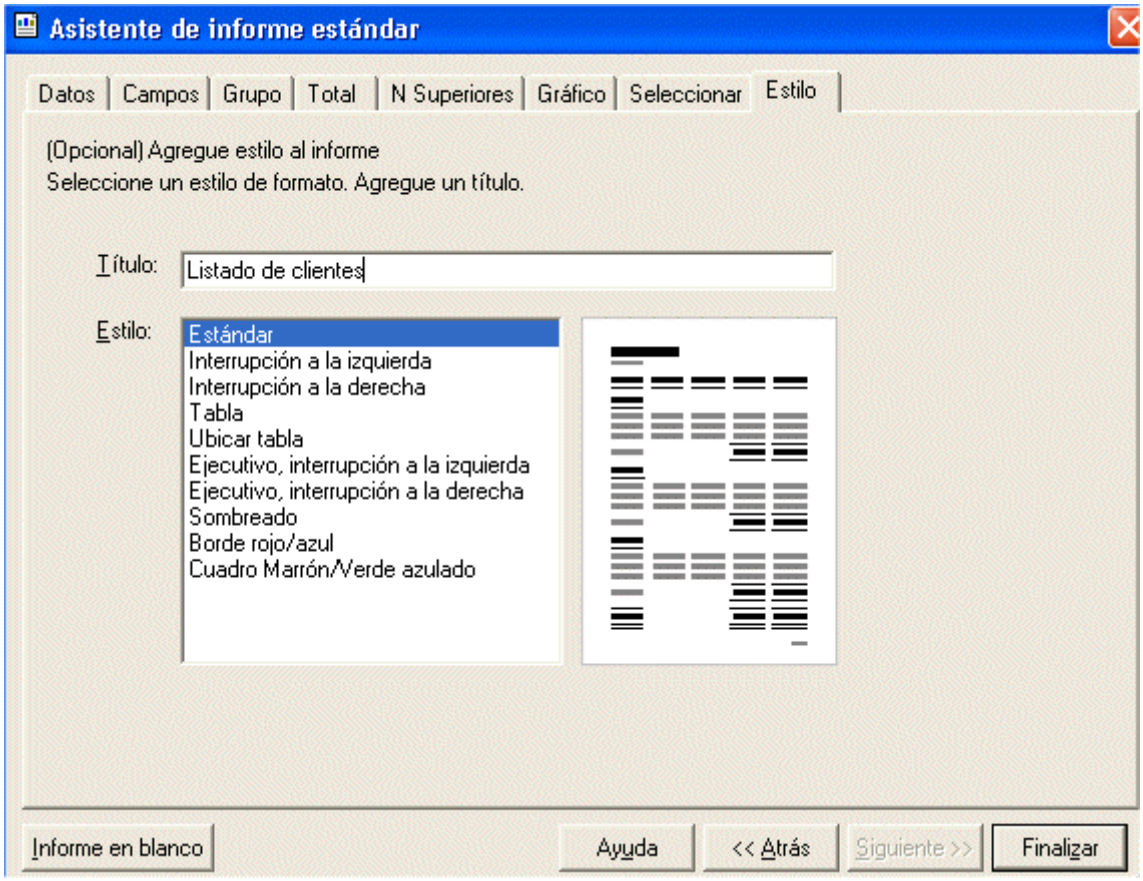


Figura 85. Introducción del título para el listado.

El título quedará situado en el encabezado del informe como un objeto de campo más, de forma que al ejecutar el listado, se mostrará el texto que hemos escrito en el asistente. Ver la Figura 86.

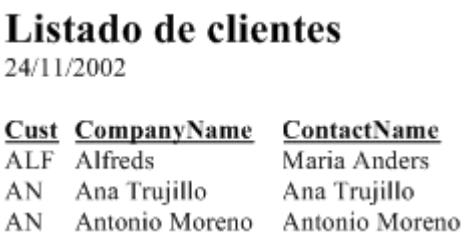


Figura 86. Título de un informe.

Supongamos ahora, que en el formulario añadimos un control TextBox en el que el usuario escribirá un nuevo título para el informe. Para cambiar por código el título del informe que hemos asignado en el asistente del informe, importaremos en primer lugar en la cabecera del archivo de código del formulario el espacio de nombres CrystalDecisions.CrystalReports.Engine, correspondiente al motor de informes de CR.

A continuación añadiremos al formulario un Button, y en el código de su evento Click, instanciaremos un objeto de la clase del informe, en este ejemplo rptClientes.

Seguidamente tomaremos el objeto informe recién creado, y obtendremos de su propiedad SummaryInfo, un objeto de dicho tipo, que contiene la información de resumen del informe.

Al objeto SummaryInfo que acabamos de obtener, le asignaremos el nuevo título del informe en su propiedad ReportTitle, y por último asignaremos el objeto informe al control visualizador para mostrar el resultado. El código correspondiente a estos pasos lo podemos ver en el Código fuente 11.

```
Imports CrystalDecisions.CrystalReports.Engine

Public Class Form1
' . . . .

Private Sub btnInforme_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnInforme.Click
    Dim oRptInforme As New rptClientes() ' instanciar el informe
    Dim oSumInfo As SummaryInfo

    oSumInfo = oRptInforme.SummaryInfo ' obtener el objeto resumen del informe
    oSumInfo.ReportTitle = Me.txtTitulo.Text ' cambiar el título
    Me.crvInforme.ReportSource = oRptInforme ' asignar informe al control
visualizador
End Sub

' . . . .
End Class
```

Código fuente 11

El resultado en tiempo de ejecución sería similar al mostrado en la Figura 87.

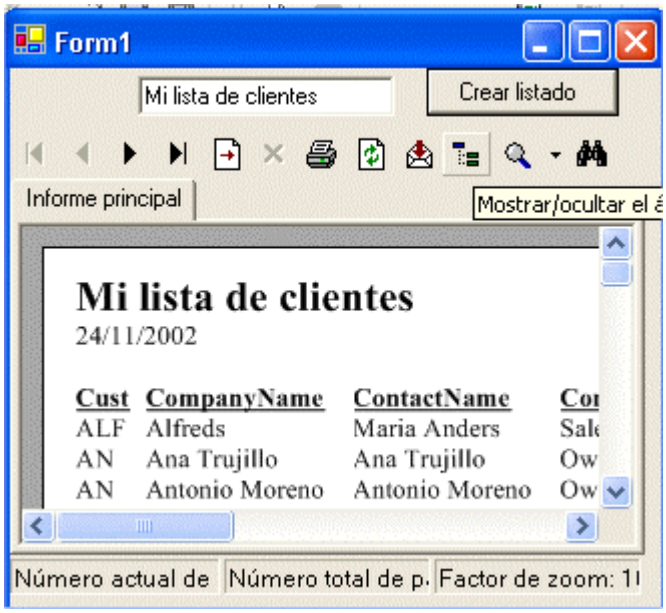


Figura 87. Título de informe introducido por código.

Selección de registros por código

Además de establecer como ya vimos en un apartado anterior, la selección de registros para el informe mediante su asistente y diseñador, en muchas ocasiones (probablemente la mayoría), necesitaremos establecer esta selección como un parámetro asignable desde el código de la aplicación, aspecto que veremos en este apartado.

Para ello crearemos un nuevo proyecto con el nombre RepSelecCodigo (para acceder al proyecto de este ejemplo hacer clic aquí), agregando un informe que inicialmente crearemos para que muestre todos los registros de la tabla Suppliers.

A continuación pasaremos al diseñador del formulario, añadiendo un control CrystalReportViewer y varios controles TextBox, Button y Label para poder establecer selecciones que muestren el informe con un filtro sobre la tabla. El objetivo en este ejemplo es ejecutar el informe con dos filtros distintos: seleccionando por el campo Country y por un intervalo del campo SupplierID. El diseño del informe se muestra en la Figura 88.

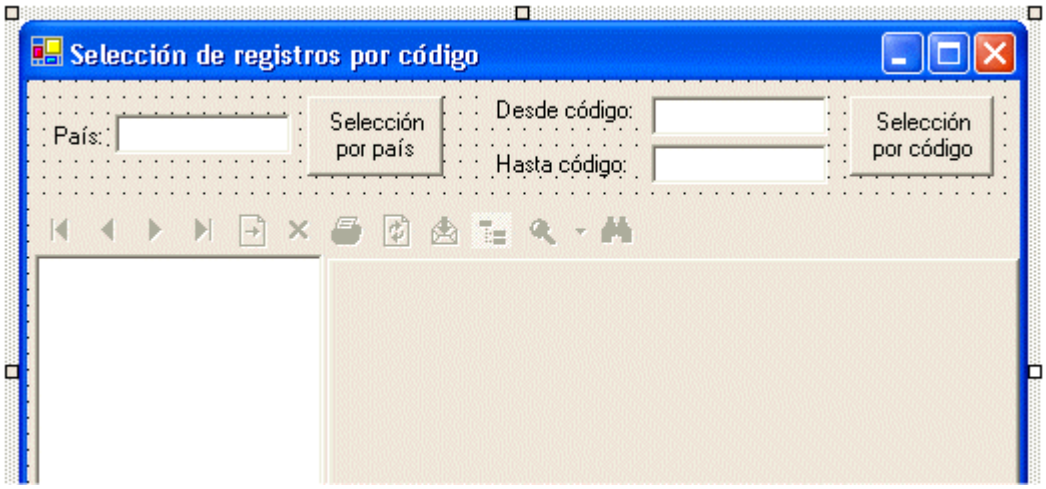


Figura 88. Formulario para seleccionar registros en el informe.

Finalmente escribiremos el código para los eventos Click de los botones del formulario. La clave en este caso se encuentra en asignar a la propiedad RecordSelectionFormula del objeto informe, una cadena con la fórmula de selección de registros que utilizará el informe al ejecutarse. El Código fuente 12 muestra el código para estos botones.

```
' selección de registros por país
Private Sub btnSelecPais_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSelecPais.Click
    ' instanciar el objeto informe
    Dim oRptPrueba As New rptPrueba()
    ' establecer la fórmula de selección de registros
    oRptPrueba.RecordSelectionFormula = "{Suppliers.Country} = '" & Me.txtPais.Text
& "'"
    ' asignar el objeto informe al control visualizador
    Me.crvInforme.ReportSource = oRptPrueba
End Sub


'-------------------------------------------------
' selección de registros por código de proveedor
Private Sub btnSelecCodigo_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSelecCodigo.Click
    ' instanciar el objeto informe
    Dim oRptPrueba As New rptPrueba()
    ' establecer la fórmula de selección de registros
    oRptPrueba.RecordSelectionFormula = "{Suppliers.SupplierID} = " &
Me.txtDesde.Text & " to " & Me.txtHasta.Text
    ' asignar el objeto informe al control visualizador
    Me.crvInforme.ReportSource = oRptPrueba
```

```
End Sub
```

Código fuente 12

A modo de muestra, en la Figura 89 podemos ver la ejecución del informe tras seleccionar los registros correspondientes a un determinado país.

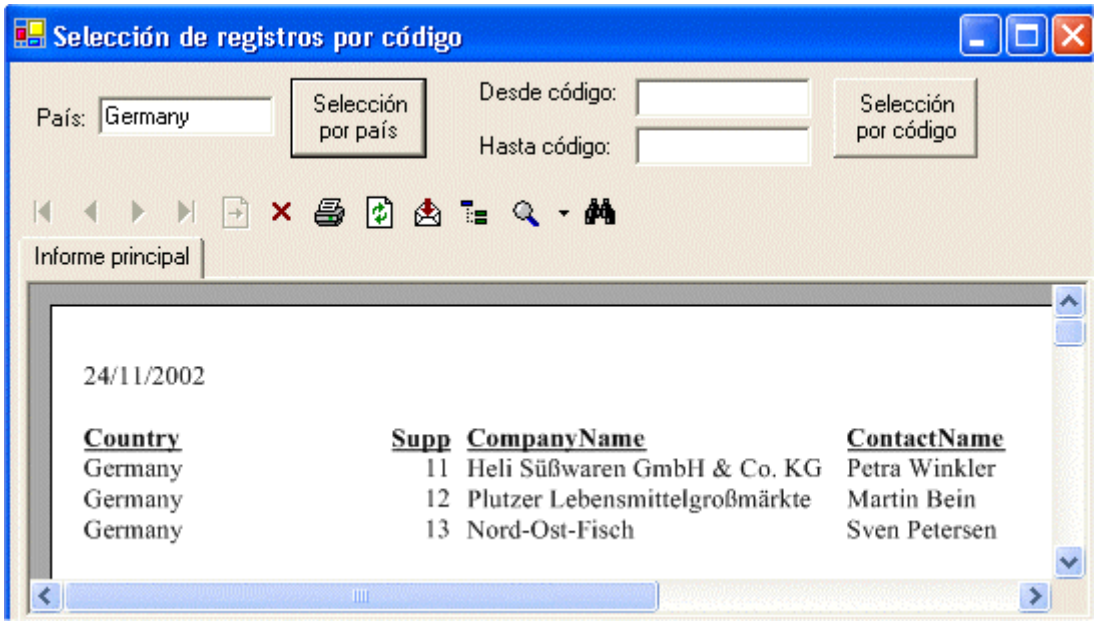


Figura 89. Informe con selección de registros por país.

Ordenación de registros por código

Al igual que sucede con la selección de registros, también podemos desde el código de la aplicación, establecer el orden de los registros en el informe.

Para ello crearemos un proyecto con el nombre RepOrdenCodigo (para acceder al proyecto de este ejemplo hacer clic aquí), que contenga un informe que muestre el contenido de la tabla Suppliers, y que inicialmente esté ordenado por el campo SupplierID. De los anteriores apartados, consulte el lector el dedicado a la creación de informes ordenados utilizando el diseñador de informes.

El propósito en este ejemplo consiste en seleccionar un campo del informe, y mostrar este ordenado por dicho campo. Para ello utilizaremos en el diseñador del formulario un ComboBox, con los siguientes valores correspondientes a nombres de campo: SupplierID, CompanyName, ContactName, City y Country. También añadiremos un control Button que al ser pulsado, visualice el informe ordenado por el campo seleccionado en el ComboBox.

Respecto al código encargado de establecer el orden de registros, en primer lugar importaremos el espacio de nombres CrystalDecisions.CrystalReports.Engine en el archivo de código del formulario, y a continuación escribiremos el código mostrado en el Código fuente 13 para el evento Click del Button.

```
Imports CrystalDecisions.CrystalReports.Engine
```

```
Public Class Form1
    Inherits System.Windows.Forms.Form
    '....

    Private Sub btnOrdenar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnOrdenar.Click
        Dim oRptPrueba As New rptPrueba()

        ' obtener el objeto definición de campo entre los campos del informe,
        ' que coincida con el que ha seleccionado el usuario en el ComboBox
        Dim oFieldDefinition As FieldDefinition
        oFieldDefinition =
oRptPrueba.Database.Tables.Item(0).Fields.Item(Me.cboOrden.Text)

        ' asignar el campo del informe seleccionado
        ' al campo de orden del informe
        oRptPrueba.DataDefinition.SortFields.Item(0).Field = oFieldDefinition

        Me.crvInforme.ReportSource = oRptPrueba
    End Sub
End Class
```

Código fuente 13

Lo que estamos haciendo en el anterior código fuente es lo siguiente: obtenemos la definición del campo del informe en un objeto de tipo FieldDefinition, según el nombre seleccionado en el ComboBox.

Para conseguir el objeto FieldDefinition, tomamos el objeto informe y vamos recorriendo su propiedad Database, dentro de esta, la colección Tables, y de las tablas elegimos una mediante la colección Item, utilizando la posición numérica.

Como sólo hay una tabla, corresponde al primer elemento de la colección, es decir, el cero. Continuamos con la colección Fields de la tabla elegida, y dentro de esta, mediante la colección Item, pasamos el nombre de campo a obtener con el valor actual del ComboBox.

Una vez obtenido el objeto con la definición de campo deberemos pasarlo al informe. Para ello, tomaremos el informe, seleccionando su propiedad DataDefinition, que como su nombre indica, contiene los miembros necesarios para establecer su definición de datos, entre ellos tomaremos la colección SortFields, que contiene los campos por los que se ordena el informe, como sólo hay uno, con la colección Item indicaremos la posición cero, y finalmente a la propiedad Field le asignaremos el objeto FieldDefinition obtenido antes. Para terminar, visualizaremos el informe en el formulario.

La Figura 90 muestra el diseño resultante de este formulario, con el informe ordenado por uno de los campos seleccionables.

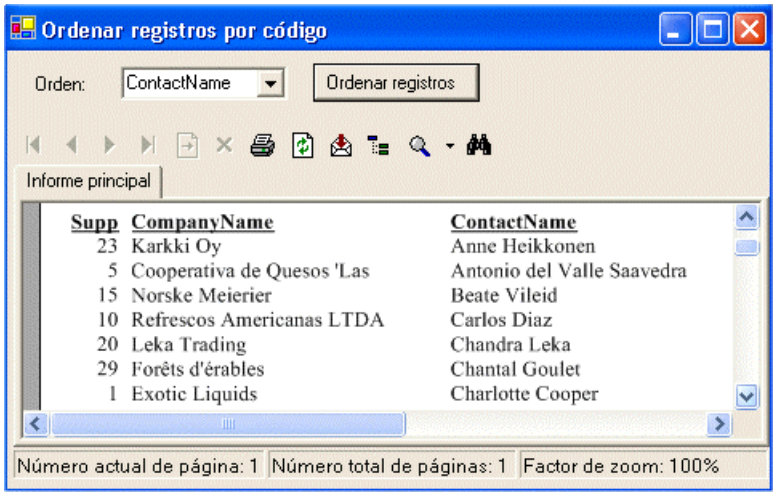


Figura 90. Informe con registros ordenados por código.

Establecer por código el grupo de registros

De modo muy similar al empleado en el apartado anterior, podemos seleccionar en un informe la manera en cómo vamos a agrupar los registros. Para ilustrar esta situación crearemos un proyecto con el nombre RepGrupoCodigo (para acceder al proyecto de este ejemplo hacer clic aquí). En este proyecto crearemos un informe que muestre los registros de la tabla Orders, y con el asistente de informes crearemos un grupo por el campo CustomerID. Para simplificar, en el caso de que el asistente genere automáticamente totales y subtotales, los eliminaremos del diseñador del informe.

A continuación insertaremos en el formulario un ComboBox con los nombres de los siguientes campos: CustomerID, EmployeeID y ShipCountry, que serán los que utilicemos para agrupar los registros. Por último añadiremos un control Button, que al ser pulsado mostrará el informe en el control visualizador del formulario, agrupado por el nombre de campo elegido en el ComboBox. El código del Button podemos verlo en el Código fuente 14.

```
' ¡¡¡ATENCIÓN!!!, debemos importar este espacio de nombres
Imports CrystalDecisions.CrystalReports.Engine

'....

Private Sub btnGenerar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnGenerar.Click
    Dim oRptPrueba As New rptPrueba()

    ' obtener el objeto definición de campo entre los campos del informe,
    ' que coincida con el que ha seleccionado el usuario en el ComboBox
    Dim oFieldDefinition As FieldDefinition
    oFieldDefinition =
oRptPrueba.Database.Tables.Item(0).Fields.Item(Me.cboGrupo.Text)

    ' asignar el campo del informe seleccionado
    ' al campo de condición de grupo del informe
    oRptPrueba.DataDefinition.Groups.Item(0).ConditionField = oFieldDefinition

    Me.crvInforme.ReportSource = oRptPrueba
End Sub
```

Código fuente 14

En la Figura 91 podemos ver el informe en ejecución agrupado por el campo ShipCountry.

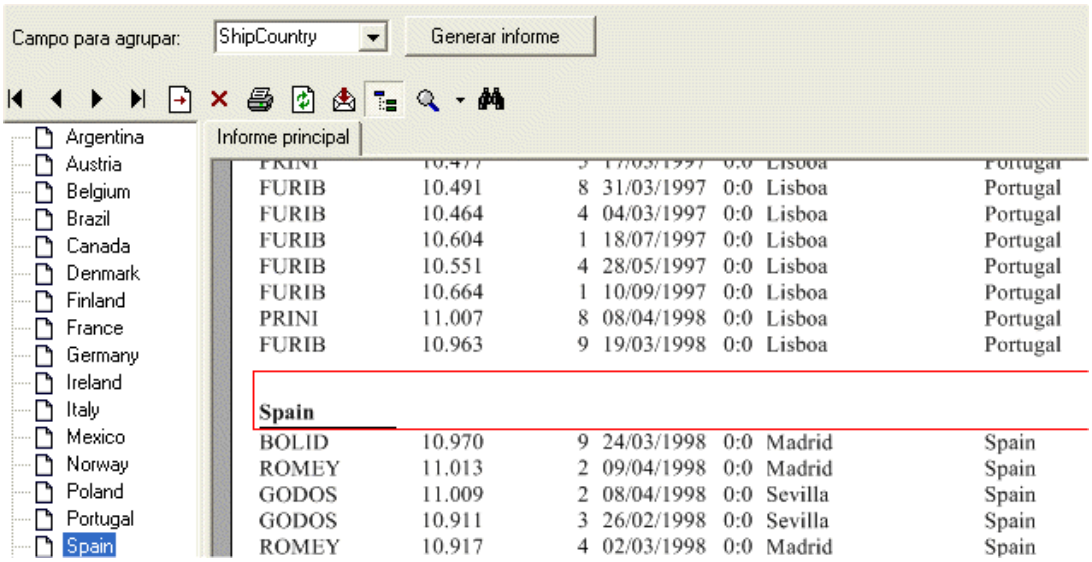


Figura 91. Informe con registros agrupados por código.

Si queremos como comúnmente se dice *rizar el rizo*, debemos tener en cuenta que las características de manipulación por código que estamos describiendo en estos apartados son combinables. Por ejemplo, en este caso concreto que nos ocupa, si además de mostrar el informe agrupado, queremos que se muestre con un filtro, por ejemplo para los registros con un determinado valor en el campo EmployeeID, vamos a añadir un TextBox adicional en el formulario, y en el código del botón, tras la selección del grupo, añadiríamos el código que muestra el Código fuente 15.

```
Private Sub btnGenerar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnGenerar.Click
    Dim oRptPrueba As New rptPrueba()

    ' obtener el objeto definición de campo entre los campos del informe,
    ' que coincida con el que ha seleccionado el usuario en el ComboBox
    Dim oFieldDefinition As FieldDefinition
    oFieldDefinition =
oRptPrueba.Database.Tables.Item(0).Fields.Item(Me.cboGrupo.Text)

    ' asignar el campo del informe seleccionado
    ' al campo de condición de grupo del informe
    oRptPrueba.DataDefinition.Groups.Item(0).ConditionField = oFieldDefinition

    ' ESTAS SON LAS NUEVAS LÍNEAS:
    ' si se ha introducido un valor para filtrar
    ' por el campo EmployeeID, establecerlo en
    ' la fórmula de selección de registros
    If Me.txtEmployeeID.Text.Length > 0 Then
        oRptPrueba.RecordSelectionFormula = "{Orders.EmployeeID} = " &
Me.txtEmployeeID.Text
    End If

    Me.crvInforme.ReportSource = oRptPrueba
End Sub
```

Código fuente 15

Veamos ahora en la Figura 92 el informe agrupado como antes por el campo ShipCountry pero filtrado para un determinado código de empleado.

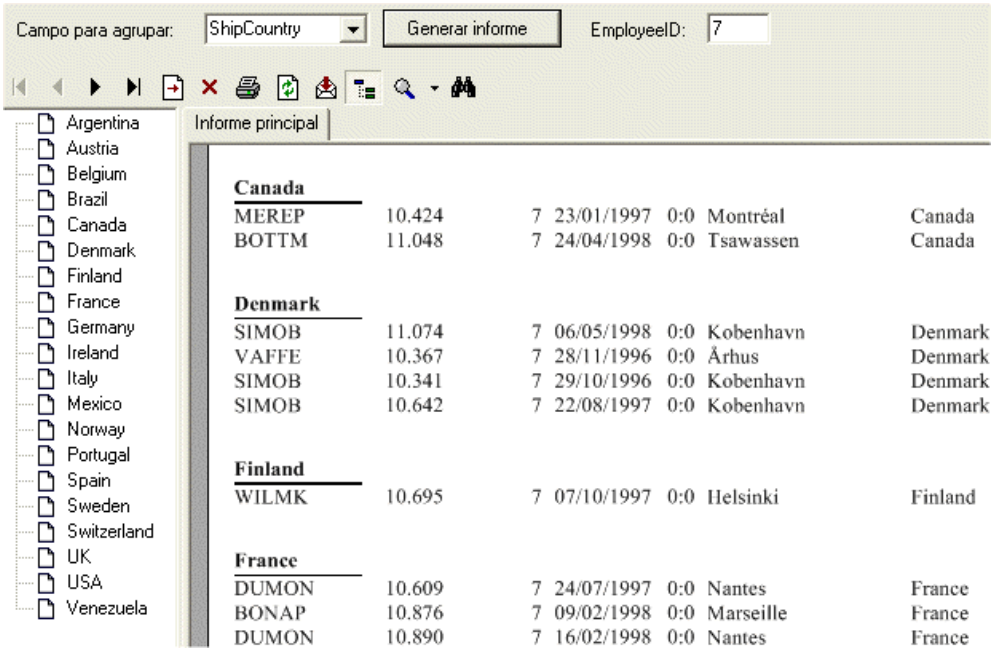


Figura 92. Informe con registros agrupados y filtrados por código.

Enviar los datos de conexión al informe por código

Cuando ejecutamos un informe para visualizarlo a través de un formulario, el motor de CR precisa la información de conexión para poder acceder al servidor de datos, y extraer los registros que se visualizarán en el listado. Dichos datos de conexión, a excepción de la contraseña del usuario en el caso de que la tenga, quedan grabados en el informe cuando este es creado con el asistente o desde la oportuna opción de conexión a base de datos en el diseñador del informe.

En los ejemplos utilizados hasta el momento, debido a que el usuario de la base de datos utilizada no tenía contraseña, el informe se ejecutaba directamente.

Para comprobar no obstante, que CR solicita este dato al ejecutar el informe, vamos a crear en nuestro servidor SQL Server una base de datos con el nombre Prueba, que tenga la típica tabla Clientes, creando también un inicio de sesión con el nombre de usuario *luis*, y la contraseña *abc*.

Seguidamente crearemos un proyecto con un informe que utilice la tabla de la base de datos que acabamos de crear. Cuando intentemos mostrar el informe en un formulario, en el evento Load( ) por ejemplo, se abrirá la caja de diálogo con los datos de conexión, solicitándonos la contraseña del usuario de la base de datos, como vemos en la Figura 93.

Al escribir la contraseña en el campo correspondiente y pulsar Finalizar, se enviarán los datos de conexión al informe, y si son correctos, se cargará el informe en el control visualizador del formulario.

Sin embargo, lo más conveniente en la mayoría de los casos será pasar por código al informe los datos de conexión para no tener que interactuar en este particular con el usuario, quien tampoco está obligado a conocer la contraseña de acceso.

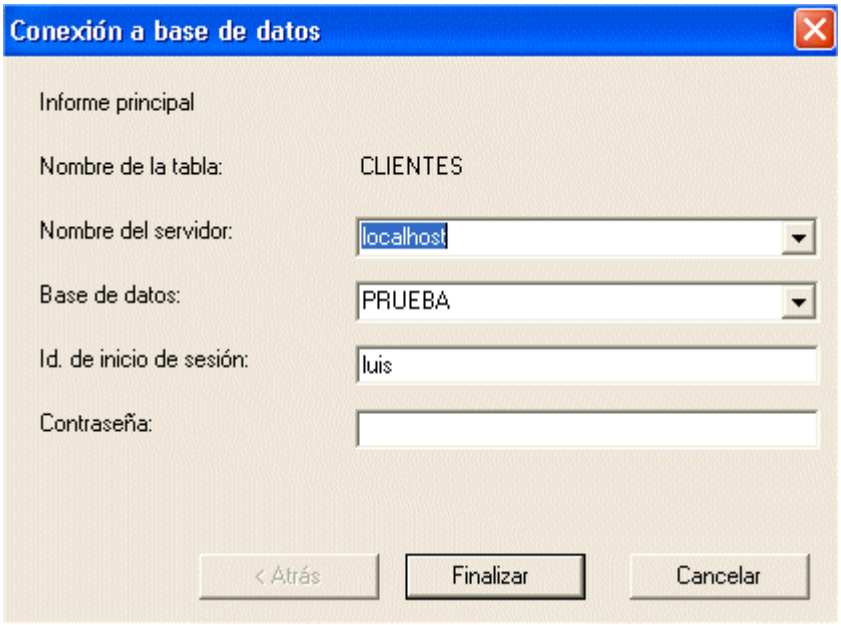


Figura 93. Caja de conexión de datos para el informe.

Partiendo del caso anterior, crearemos un proyecto con el nombre RepConexCodigo (para acceder al proyecto de este ejemplo hacer clic aquí), y en el evento Load( ) del formulario, tendremos que codificar algunas líneas adicionales, que hagan uso de los objetos para conexión con informe: ConnectionInfo, TableLogOnInfo, etc. El resultado lo podemos ver en el Código fuente 16 con explicaciones complementarias dentro del propio código.

```
' importar espacios de nombres
Imports CrystalDecisions.CrystalReports.Engine
Imports CrystalDecisions.Shared

Public Class Form1
       '....
    Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        ' declarar objetos
        Dim oRptPrueba As rptPrueba
        Dim oConexInfo As ConnectionInfo
        Dim oListaTablas As Tables
        Dim oTabla As Table
        Dim oTablaConexInfo As TableLogOnInfo

        ' instanciar objeto para guardar datos de conexión
        oConexInfo = New ConnectionInfo()
        oConexInfo.ServerName = "localhost"
        oConexInfo.DatabaseName = "PRUEBA"
        oConexInfo.UserID = "luis"
        oConexInfo.Password = "abc"

        ' instanciar objeto informe
        oRptPrueba = New rptPrueba()

        ' obtener la colección de tablas del informe
        oListaTablas = oRptPrueba.Database.Tables

        ' por cada tabla del informe...
        For Each oTabla In oListaTablas
            ' ...obtener el objeto con los datos de conexión
```

```
            oTablaConexInfo = oTabla.LogOnInfo
            ' asignar el objeto con datos de conexión
            ' que hemos creado
            oTablaConexInfo.ConnectionInfo = oConexInfo
            ' aplicar cambios de conexión a la tabla
            oTabla.ApplyLogOnInfo(oTablaConexInfo)
        Next

        ' visualizar informe en un control CrystalReportViewer
        Me.crvInforme.ReportSource = oRptPrueba
    End Sub
End Class
```

Código fuente 16

Como se puede observar en el código anterior, esta técnica requiere que especifiquemos los datos de conexión para cada una de las tablas que componen el informe y que se encuentran en la colección Tables de la propiedad Database del objeto informe.

Modificar campos de fórmula y texto por código

Una vez creada una fórmula y añadida al diseñador del informe, como ocurre con otros elementos del mismo, es posible su modificación por código en tiempo de ejecución. De la misma manera podemos hacer con los campos de texto de un informe, cambiando dinámicamente el literal que inicialmente se les asignó en diseño.

Como demostración de esta característica de CR, vamos a crear un proyecto con el nombre RepFormulaCodigo (para acceder al proyecto de este ejemplo hacer clic aquí), añadiendo un informe que obtenga los registros de la tabla Order Details.

A continuación vamos a crear dos fórmulas para el informe con los nombres Impuesto y TotalLin, con la particularidad de que en la ventana del editor de fórmula no escribiremos código, vamos a dejarlas vacías.

Una vez creadas, añadiremos estos campos de fórmula al informe como vemos en la Figura 94.

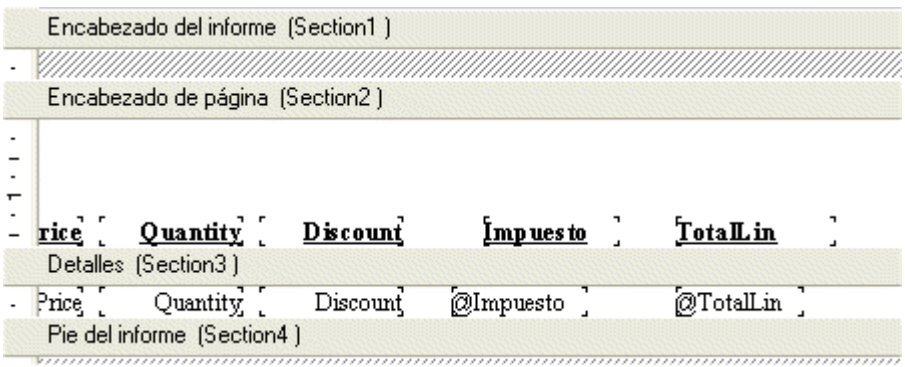


Figura 94. Informe con campos de fórmula.

Seguidamente pasaremos al formulario, en el que añadiremos dos controles ComboBox, uno con el nombre cboTipoImp, y que tenga en su lista los siguientes valores: IVA, ESPECIAL y ADUANAS. El otro control tendrá el nombre cboTipoTotal, y en su lista tendrá los valores: NORMAL, DOBLE y MITAD. Según el valor que seleccionemos en estos controles crearemos la fórmula que calcule el tipo

de impuesto a partir del campo UnitPrice, y por otro lado el total, combinando los campos UnitPrice y Quantity. También modificaremos el campo de texto que sirve como título de columna para estas fórmulas en el informe.

Para ejecutar el informe, una vez realizadas las selecciones en las listas, añadiremos un control Button, en el que escribiremos las líneas que se muestran en el Código fuente 17.

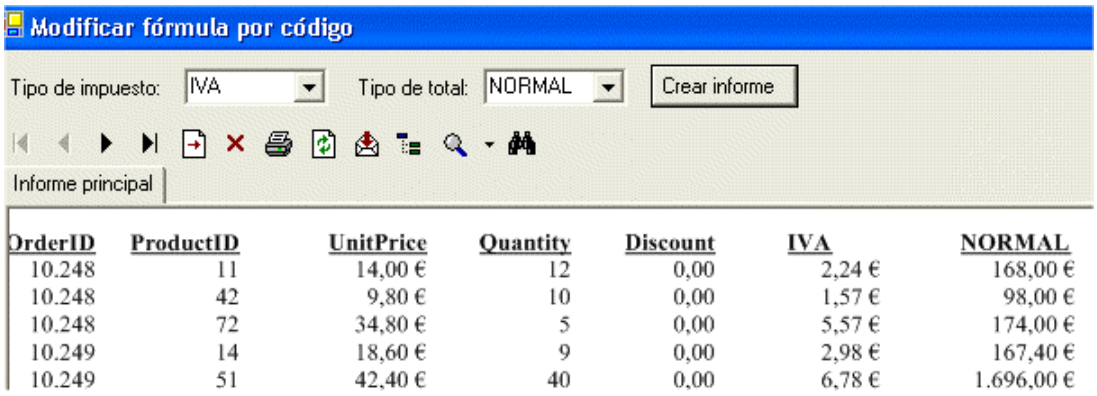
```
Imports CrystalDecisions.CrystalReports.Engine

Public Class Form1
    '....
    Private Sub btnCrear_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCrear.Click
        Dim oRptPrueba As New rptPrueba()
        Dim toImpuesto As TextObject
        Dim toTotal As TextObject

        ' obtener uno de los objetos campo de texto del informe
        ' y cambiarle el texto que visualiza
        toImpuesto = oRptPrueba.ReportDefinition.ReportObjects.Item("txtImpuesto")
        toImpuesto.Text = Me.cboTipoImp.Text

        ' en función del tipo de impuesto seleccionado en el ComboBox
        ' asignar el valor a la fórmula
        Select Case Me.cboTipoImp.Text
            Case "IVA"
                oRptPrueba.DataDefinition.FormulaFields.Item("Impuesto").Text =
"{Order_Details.UnitPrice} * 0.16"

            Case "ESPECIAL"
                oRptPrueba.DataDefinition.FormulaFields.Item("Impuesto").Text =
"{Order_Details.UnitPrice} / 2"

            Case "ADUANAS"
                oRptPrueba.DataDefinition.FormulaFields.Item("Impuesto").Text =
"{Order_Details.UnitPrice} * 2"
        End Select

        ' cambiar valor del otro campo de texto y fórmula correspondiente
        toTotal = oRptPrueba.ReportDefinition.ReportObjects.Item("txtTotalLin")
        toTotal.Text = Me.cboTipoTotal.Text
        Select Case Me.cboTipoTotal.Text
            Case "NORMAL"
                oRptPrueba.DataDefinition.FormulaFields.Item("TotalLin").Text =
"{Order_Details.UnitPrice} * {Order_Details.Quantity}"

            Case "DOBLE"
                oRptPrueba.DataDefinition.FormulaFields.Item("TotalLin").Text =
"({Order_Details.UnitPrice} * {Order_Details.Quantity}) * 2"

            Case "MITAD"
                oRptPrueba.DataDefinition.FormulaFields.Item("TotalLin").Text =
"({Order_Details.UnitPrice} * {Order_Details.Quantity}) / 2"
        End Select

        Me.crvInforme.ReportSource = oRptPrueba
    End Sub
End Class
```

Código fuente 17

Como puede observar el lector por el anterior código fuente, para obtener uno de los objetos campos de texto del informe, hacemos uso de la propiedad ReportDefinition del objeto informe, que contiene la colección ReportObjects, mediante la que manipulamos los objetos del informe, en este caso los campos de texto.

Para manejar las fórmulas tenemos que hacer uso de la propiedad DataDefinition del informe, y utilizar su colección FormulaFields, accediendo de esta manera a los campos de fórmula añadidos al informe.

En la Figura 95 podemos ver el informe en ejecución con las columnas correspondientes a las fórmulas y los títulos asignados por código.



OrderID	ProductID	UnitPrice	Quantity	Discount	IVA	NORMAL
10.248	11	14,00 €	12	0,00	2,24 €	168,00 €
10.248	42	9,80 €	10	0,00	1,57 €	98,00 €
10.248	72	34,80 €	5	0,00	5,57 €	174,00 €
10.249	14	18,60 €	9	0,00	2,98 €	167,40 €
10.249	51	42,40 €	40	0,00	6,78 €	1.696,00 €

Figura 95. Informe con fórmulas y campos de texto modificados por código.

Otra técnica de manipulación de las fórmulas de un objeto informe, consiste en recorrer la colección que contiene las definiciones de campos de las fórmulas, que se encuentra en la clase FormulaFieldDefinitions; cada elemento de esta colección contiene un objeto del tipo FormulaFieldDefinition, de manera que dentro de un bucle, podemos manipular los miembros de este objeto para cambiar el código de algunas o todas las fórmulas de un informe.

Como ejemplo de este punto, añadiremos un nuevo botón al informe, que al ser pulsado, recorra esta colección y asigne a cada fórmula del informe las instrucciones para efectuar un determinado cálculo. El código de este botón lo vemos en el Código fuente 18.

```
Imports CrystalDecisions.CrystalReports.Engine

Public Class Form1
    '....
    Private Sub btnRecorrer_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnRecorrer.Click
        ' declarar variables para guardar la lista de definiciones
        ' de campos de fórmula
        ' y cada campo de definición de fórmula independiente
        Dim oFormulaFDefiniciones As FormulaFieldDefinitions
        Dim oFormulaDefCampo As FormulaFieldDefinition
        ' instanciar objeto informe
        Dim oRptPrueba As New rptPrueba()

        ' obtener de la definición de datos del objeto informe
        ' la colección de campos de fórmula
        oFormulaFDefiniciones = oRptPrueba.DataDefinition.FormulaFields

        ' recorrer la colección de campos de fórmula
```

```
        For Each oFormulaDefCampo In oFormulaFDefiniciones
            ' tomar cada campo de fórmula y asignarle
            ' el código de la fórmula
            Select Case oFormulaDefCampo.FormulaName
                Case "{@Impuesto}"
                    oFormulaDefCampo.Text = "{Order_Details.UnitPrice} + 1000"

                Case "{@TotalLin}"
                    oFormulaDefCampo.Text = "{Order_Details.UnitPrice} + 5000"

            End Select
        Next

        Me.crvInforme.ReportSource = oRptPrueba
    End Sub
    '....
End Class
```

Código fuente 18

La Figura 96 muestra un ejemplo de la ejecución.

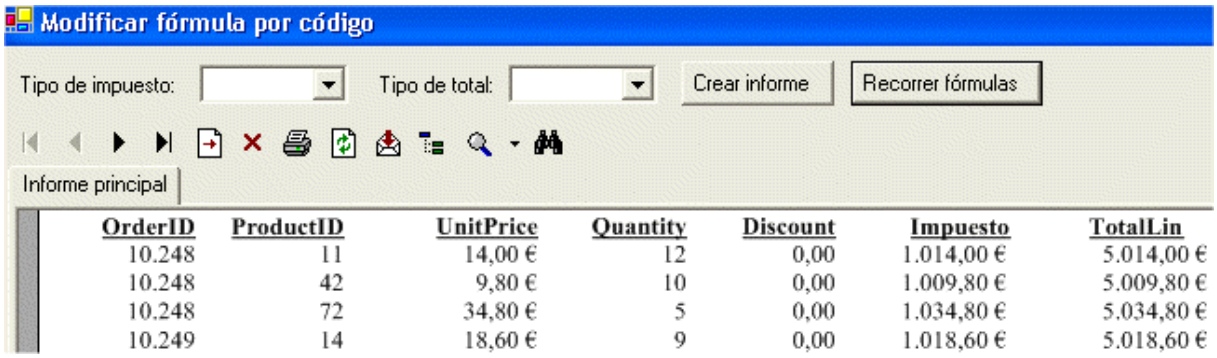


Figura 96. Informe resultante de manipular la colección de fórmulas.

Impresión del informe desde código

Para imprimir un informe por código sin utilizar el control CrystalReportViewer, llamaremos al método PrintToPrinter( ) del objeto informe, cuya sintaxis se muestra en el Código fuente 14.

```
PrintToPrinter(Copias, Intercalación, PrimeraPag, ÚltimaPag)
```

Código fuente 19

Los parámetros de este método son los siguientes:

- **Copias**. Integer. Número de copias a imprimir.
- **Intercalación**. Boolean. Indica si las páginas se van a intercalar.
- **PrimeraPag**. Integer. Primera página a imprimir.

- **ÚltimaPag**. Integer. Última página a imprimir.

Adicionalmente, es posible configurar los aspectos relacionados con la impresión del informe a través de su propiedad PrintOptions, que contiene un objeto de este mismo nombre, mediante el que podremos seleccionar el tamaño de papel, orientación del mismo, impresora, etc.

El Código fuente 20 muestra un ejemplo de configuración e impresión de un objeto informe.

```
' crear un objeto informe
Dim oInforme As New rptPrueba()

' configurar las opciones del informe
oInforme.PrintOptions.PaperSize = PaperSize.PaperLetter ' tamaño de papel
oInforme.PrintOptions.PaperSource = PaperSource.Upper ' bandeja
oInforme.PrintOptions.PrinterName = "DptoAdmin" ' impresora

' imprimir el informe
oInforme.PrintToPrinter(1, False, 0, 0)
```

Código fuente 20

Utilizar un procedimiento almacenado para obtener los registros

Los orígenes de datos tradicionales (léase SGBDR) no son las únicas fuentes de datos que podemos utilizar en Crystal Reports para .NET. Como explicaremos en los próximos apartados, podemos utilizar elementos ya habituales como los procedimientos almacenados, y otros más novedosos, como los DataSets, para llenar nuestros informes de contenido.

En el caso que nos ocupa, vamos a utilizar un procedimiento almacenado. Para desarrollar el ejemplo, crearemos un proyecto con el nombre RepProcAlmac (para acceder al proyecto de este ejemplo hacer clic aquí), y agregaremos un informe que inicialmente muestre la tabla Suppliers, pero sólo con los campos SupplierID, CompanyName, City y Country.

Si ejecutamos en este momento el proyecto, cargando el informe en un control visualizador en el formulario, los registros aparecerán por el orden predeterminado de la tabla Suppliers, que es el campo SupplierID

Seguiremos con la creación, bien desde el Administrador corporativo o el Analizador de consultas de SQL Server, del procedimiento almacenado que usará el informe, al que daremos el nombre ProveedPorPais, y cuyo contenido se muestra en el Código fuente 21.

```
CREATE PROCEDURE ProveedPorPais
AS
SELECT SupplierID, CompanyName, City, Country
FROM Suppliers
ORDER BY Country
```

Código fuente 21

A continuación hemos de verificar que en la configuración del diseñador de informes podemos utilizar procedimientos almacenados, para lo que haremos clic derecho en el informe, y seleccionaremos *Diseñador + Especificaciones predeterminadas* en el menú contextual.

El cuadro de diálogo que se muestra seguidamente, nos permite establecer valores genéricos para todos los informes creados con el diseñador de CR. En nuestro caso concreto, haremos clic en la pestaña Base de datos, y marcaremos la casilla Procedimientos almacenados, aceptando por último esta ventana.

El siguiente paso consistirá en hacer nuevamente clic derecho en el diseñador de informes, seleccionando esta vez el menú *Base de datos + Establecer ubicación*. En la ventana Establecer ubicación (ver la Figura 97), desplegaremos el ComboBox Origen de datos actual, y seleccionaremos la tabla Suppliers, que es la que hasta el momento usa el informe. A continuación, desplegaremos los nodos del panel Reemplazar con, hasta localizar el procedimiento almacenado que hemos creado. La ruta de nodos que tendremos que ir expandiendo será similar a la siguiente: OLE DB (ADO) + localhost + Northwind + dbo + Procedimientos almacenados + ProveedPorPais. Por último en este paso, pulsaremos el botón Reemplazar, para actualizar en el informe el origen de datos, y cerraremos esta ventana.

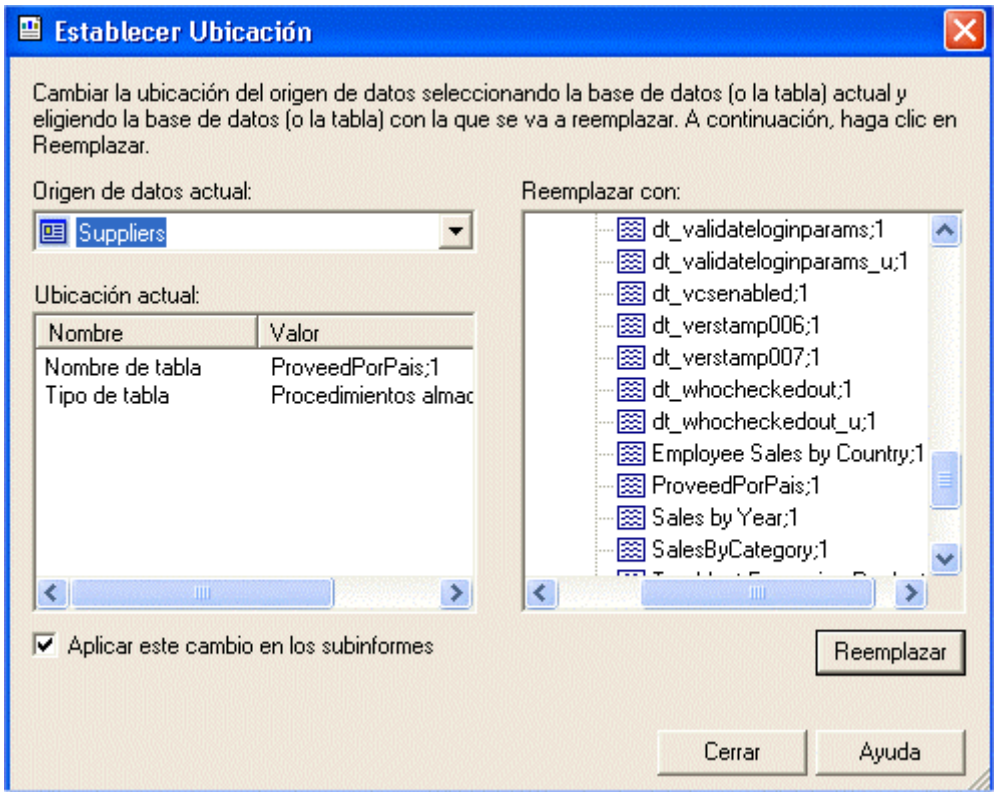


Figura 97. Estableciendo un procedimiento almacenado como origen de datos para el informe.

Tras dar estos pasos, volveremos a ejecutar el informe, comprobando ya, que los registros de la tabla se muestran ordenados por el campo Country, fruto de la ejecución del procedimiento almacenado.

Paso de parámetros a un informe

Como acabamos de comprobar en el apartado anterior, el uso de procedimientos almacenados nos abre una nueva vía en la obtención de datos para nuestros informes, pero también es cierto que en la mayor parte de las ocasiones, dichos procedimientos contienen parámetros, que les confieren mayor flexibilidad, al poder obtener conjuntos de resultados distintos, en función de los valores pasados al procedimiento. En este apartado veremos cómo podemos pasar valores a un procedimiento almacenado utilizado por un informe, a través de los campos de parámetros de CR.

Como es tónica habitual, crearemos un proyecto con el nombre RepParam (para acceder al proyecto de este ejemplo hacer clic aquí), añadiendo un informe que utilice la tabla Suppliers, para posteriormente, configurar este informe del modo explicado en el apartado anterior, de manera que utilice el procedimiento almacenado del Código fuente 22, que previamente deberemos haber creado en nuestro servidor SQL Server.

```
CREATE PROCEDURE ProveedDeUnPais
@NombrePais varchar(50)
AS
SELECT SupplierID, CompanyName, City, Country
FROM Suppliers
WHERE Country=@NombrePais
ORDER BY City
```

Código fuente 22

Tras establecer en este ejemplo, el procedimiento almacenado como nueva ubicación para los datos, dado que este procedimiento tiene un parámetro, el motor de CR creará automáticamente un campo de parámetro, que podemos ver en la ventana Explorador de campos, expandiendo el nodo correspondiente a estos elementos del informe. Ver la Figura 98.

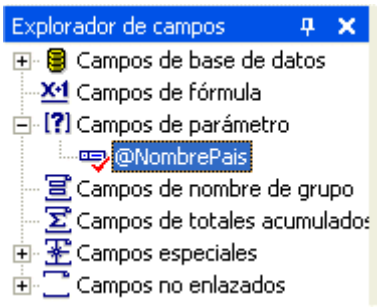


Figura 98. Campos de parámetro del informe.

Si en el estado actual del informe, lo ejecutamos cargándolo en un control visualizador, dentro del evento Load( ) del formulario, el motor de CR detectará que necesita un valor para pasarlo a su vez al procedimiento almacenado y que le devuelva los registros correspondientes. Como actualmente no pasamos dicho valor, será Crystal Reports quien lo pedirá a través de la ventana de la Figura 99. En el campo reservado para el valor lo introduciremos, y tras aceptar la ventana se mostrará el informe, acorde con el parámetro enviado.

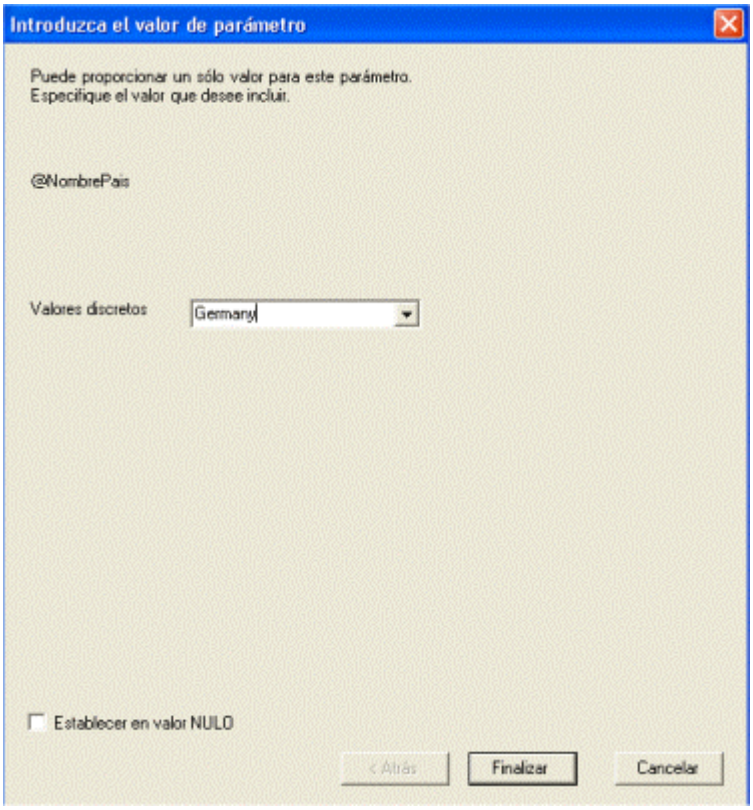


Figura 99. Ventana para introducción de parámetros en el informe.

Evidentemente, esta no es la manera idónea de proceder en una situación como la que nos encontramos, ya que lo que nos interesa realmente, es poder pasar por código los parámetros al informe. Aspecto que describimos a continuación.

Para permitir al usuario la introducción del valor para el parámetro, añadiremos en el formulario un TextBox para escribir dicho valor, y un Button para ejecutar el informe. En el código del control Button escribiremos la lógica a seguir para este proceso, que vemos en el Código fuente 23, en el que se han incluido los comentarios oportunos para las clases a utilizar.

```
' en esta ocasión debemos utilizar este espacio de nombres
Imports CrystalDecisions.Shared

Public Class Form1
    '....
    Private Sub btnGenerar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnGenerar.Click
        ' objeto para contener una colección de valores de parámetros
        Dim pvValoresParametros As New ParameterValues()
        ' objeto para almacenar un valor de un parámetro
        Dim parPais As New ParameterDiscreteValue()

        Dim oRptPrueba As New rptPrueba()

        ' asignar el valor al parámetro
        parPais.Value = Me.txtPais.Text
        ' añadir el parámetro a la colección de parámetros
        pvValoresParametros.Add(parPais)
        ' tomar del informe, el parámetro que necesitemos,
        ' a través de su colección de campos de parámetros,
        ' y aplicar a dicho parámetro los valores de los
```

```
        ' parámetros que están en la colección (en este caso sólo uno)
oRptPrueba.DataDefinition.ParameterFields("@NombrePais").ApplyCurrentValues(pvValor
esParametros)

        Me.crvInforme.ReportSource = oRptPrueba
    End Sub
End Class
```

Código fuente 23

El resultado en tiempo de ejecución de este ejemplo, se muestra en la Figura 100.

Figura 100. Ejecución de informe con parámetros.

Emplear un DataSet en el informe para obtener los datos

La nueva versión de Crystal Reports, ha sido desarrollada para obtener provecho de las principales novedades que aporta el nuevo modelo de objetos para el acceso a datos de la plataforma .NET Framework: ADO.NET.

Es posible crear un DataSet en un proyecto, y conseguir que este objeto sea el proveedor de los datos para un informe de CR, como explicaremos a continuación.

Para ello, crearemos un nuevo proyecto con el nombre RepDataSet (para acceder al proyecto de este ejemplo hacer clic aquí), y agregaremos al mismo un DataSet con el nombre dsNorthwind mediante la opción de menú *Proyecto + Agregar nuevo elemento*. Ver la Figura 101.

A continuación, abriremos la ventana Explorador de servidores, y crearemos una conexión para la base de datos Northwind en el caso de que no la tengamos previamente creada.

Como siguiente paso, expandiremos de la conexión que hemos creado, el nodo que contiene las tablas, y arrastraremos la tabla Suppliers hasta la ventana de diseño del DataSet, que nos mostrará gráficamente, la estructura de esta tabla, como vemos en la Figura 102.

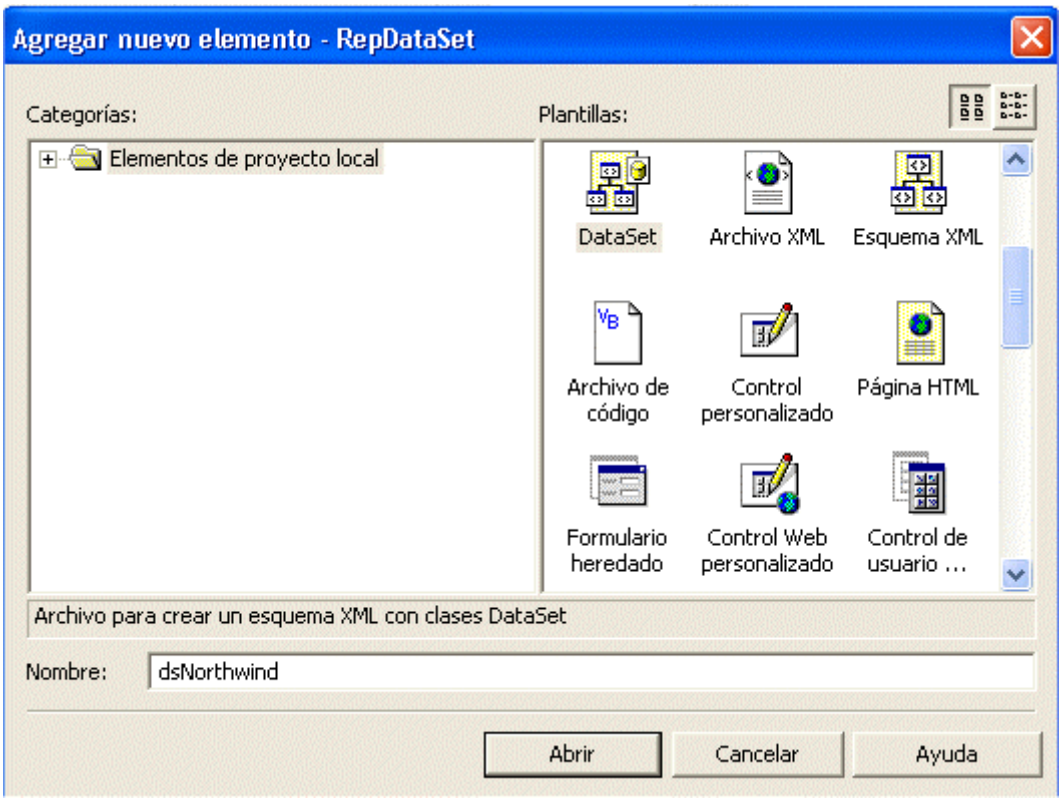


Figura 101. Agregar un nuevo DataSet al proyecto.

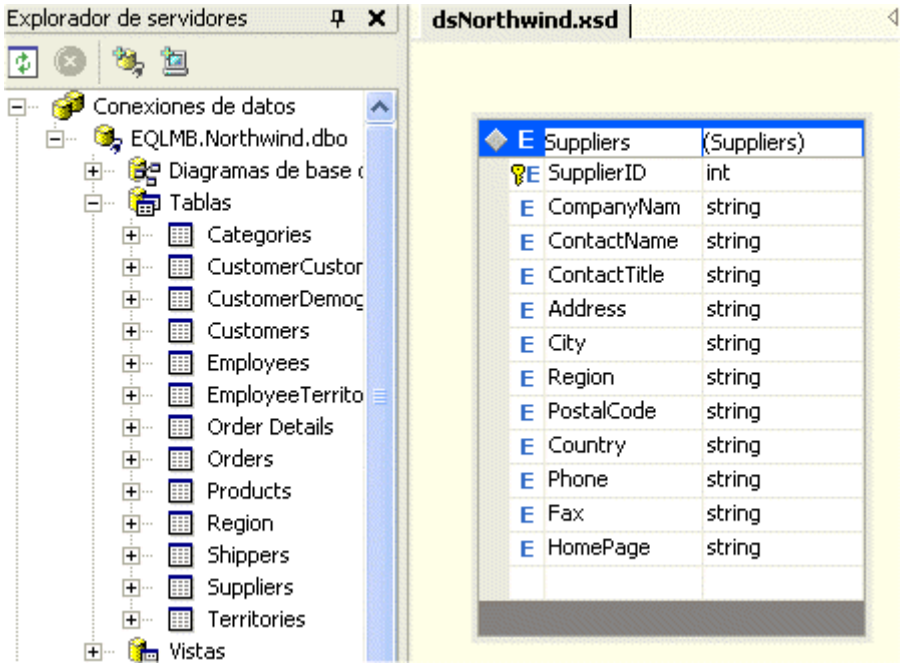


Figura 102. Creación gráfica de un DataSet.

Seguidamente añadiremos un informe con el nombre rptPrueba al proyecto, seleccionando en la ventana inicial del asistente de informes, que vamos a crearlo como un informe en blanco, y finalizando en ese momento el asistente.

Para establecer el origen de los datos para el informe, haremos clic derecho en su diseñador, seleccionando del menú contextual la opción *Base de datos + Agregar base de datos*. Esta acción mostrará un cuadro de diálogo, en el que iremos expandiendo, desde el nodo Datos del proyecto, hasta llegar al DataSet que hemos creado anteriormente. Seleccionaremos la tabla contenida en el DataSet, y la añadiremos al informe, aceptando esta ventana. Ver la Figura 103.

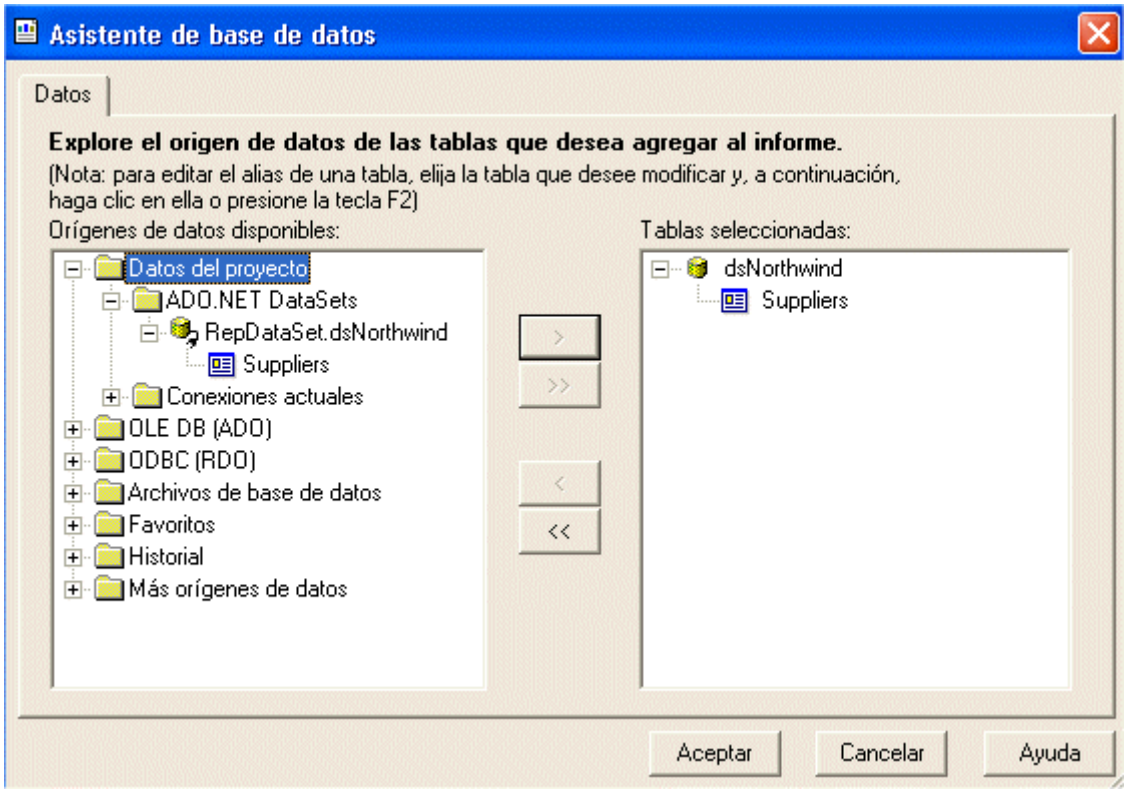


Figura 103. Selección del DataSet como origen de datos para el informe.

En el siguiente paso iremos a la ventana Explorador de campos, y expandiendo el nodo Campos de base de datos, llegaremos a la tabla Suppliers, desde la que arrastraremos varios campos hasta el diseñador del informe.

A continuación nos situaremos en el diseñador del formulario del proyecto, y desde la conexión que hemos creado antes en el Explorador de servidores, arrastraremos y soltaremos en el formulario la tabla Suppliers. Esto creará automáticamente un objeto SqlConnection y un SqlDataAdapter en el panel de elementos ocultos del formulario, a los que cambiaremos sus nombres por defecto, por unos más adecuados, en este caso conNorthwind y daNorthwind respectivamente.

Después haremos clic derecho sobre el objeto DataAdapter, seleccionando la opción de menú contextual *Generar conjunto de datos*, y en el cuadro de diálogo de este mismo nombre, seleccionaremos el DataSet que habíamos creado con anterioridad, y que ya se encuentra marcado por defecto, como vemos en la Figura 104.

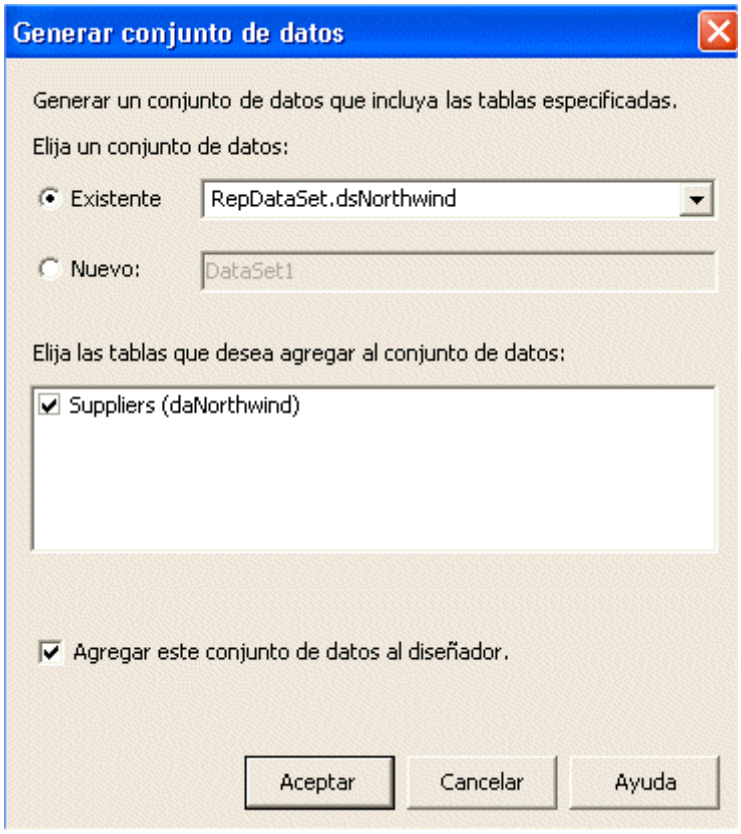


Figura 104. Generar DataSet a partir de un DataAdapter.

El resultado será la generación de un nuevo objeto para el formulario, correspondiente al DataSet, con el nombre DsNorthwind1, que también será situado en el panel de elementos ocultos del formulario. Ver la Figura 105.

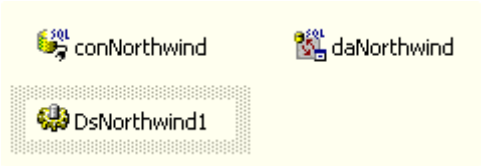


Figura 105. Objetos de datos en el panel de elementos ocultos del formulario.

Por último, escribiremos en el evento Load( ) del formulario, el código que crea una instancia del informe, rellena el DataSet mediante el DataAdapter, y asigna el DataSet al objeto informe, para finalmente, visualizarlo en un control CrystalReportViewer, insertado en el formulario del proyecto. Ver Código fuente 24.

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    ' instanciar el objeto informe
    Dim oRptPrueba As New rptPrueba()

    ' utilizar el DataAdapter para rellenar el DataSet
    Me.daNorthwind.Fill(Me.DsNorthwind1)

    ' tomar el objeto informe, y establecerle
```

```
    ' como fuente de datos, el DataSet que
    ' acabamos de rellenar de datos
    oRptPrueba.SetDataSource(Me.DsNorthwind1)

    ' cargar el informe en el control visualizador
    Me.crvInforme.ReportSource = oRptPrueba
End Sub
```

Código fuente 24

Crear un informe a partir de un archivo XML y su esquema correspondiente

En todos los ejemplos vistos hasta el momento, los informes han obtenido los datos a mostrar desde una base de datos, bien directamente a través del proveedor OLE DB correspondiente, o utilizando un DataSet intermedio.

Como ya sabemos, la arquitectura de acceso a datos con ADO.NET se sustenta principalmente en el paradigma de XML. Uno de los principales objetos de este modelo, DataSet, está basado en XML, y dispone de características que le permiten manipular la información en este formato. Este hecho nos puede llevar a la siguiente conclusión: si un informe puede obtener los datos a partir de un DataSet conectado a una fuente de datos tradicional o SGBDR, también puede utilizar como origen de datos un archivo XML, partiendo del adecuado esquema XSD.

Vamos a plantear el siguiente escenario de trabajo: supongamos que la concejalía de transportes de una ciudad, encarga un estudio de uso del transporte público, para lo que se diseña un software que será utilizado en unos dispositivos móviles por los encuestadores que realicen el mencionado estudio.

Por cuestiones de diseño de la aplicación, los datos obtenidos por los usuarios no pueden ser directamente depositados en una base de datos, pero es necesario tener listados informativos de cada encuestador, incluso antes de realizar el volcado de información a la base de datos.

Ante esta situación, una posible solución sería que la información introducida en la aplicación fuera grabada en un archivo XML, que dada su versatilidad al ser usado con ADO.NET, nos va a permitir generar los informes en Crystal Reports directamente de dicho archivo, sin necesidad de haber traspasado su contenido a la base de datos de la aplicación central.

Para este ejemplo sólo nos vamos a centrar en resolver la parte dedicada a la generación de los informes. Crearemos por lo tanto un nuevo proyecto con el nombre RepEsquema (para acceder al proyecto de este ejemplo hacer clic aquí), e iremos realizando los pasos que se describen a continuación.

En primer lugar, debemos diseñar un DataSet, que nos genere el correspondiente esquema basado en XML, para lo que mediante la opción de agregar un nuevo elemento de VS.NET, agregaremos un elemento de este tipo con el nombre dsViajes, que tenga la estructura mostrada en la Figura 106.

De esta forma obtendremos lo que se denomina un DataSet con tipo o *Typed DataSet*.

Otro modo más rudimentario de crear el esquema del DataSet, pasaría por escribir dicho esquema en un archivo de texto, por ejemplo con el Bloc de notas, grabarlo con extensión .XSD, y añadirlo posteriormente desde VS.NET al proyecto. Sin embargo, gracias al diseñador de esquemas del IDE no es necesario complicarnos con la escritura en XML del esquema, además de que podemos visualizar en todo momento el código generado con la vista en XML de que dispone este diseñador.

Figura 106. Estructura del DataSet creado manualmente.

A continuación añadiremos un informe al proyecto, y en el apartado del asistente dedicado a la conexión de datos, expandiremos desde el nodo *Más orígenes de datos*, hasta llegar a ADO.NET (XML), que abrirá una caja de diálogo en la que se nos permite especificar un archivo XML externo al proyecto, o una clase DataSet del proyecto que contenga la estructura de datos.

Como muestra la Figura 107, en este caso conectaremos con el DataSet que hemos generado, marcando la casilla *Clase de datos de Visual Studio*, y eligiendo el DataSet de la lista desplegable *Nombre de clase de datos de Visual Studio*. Pulsaremos Finalizar para confirmar este paso.

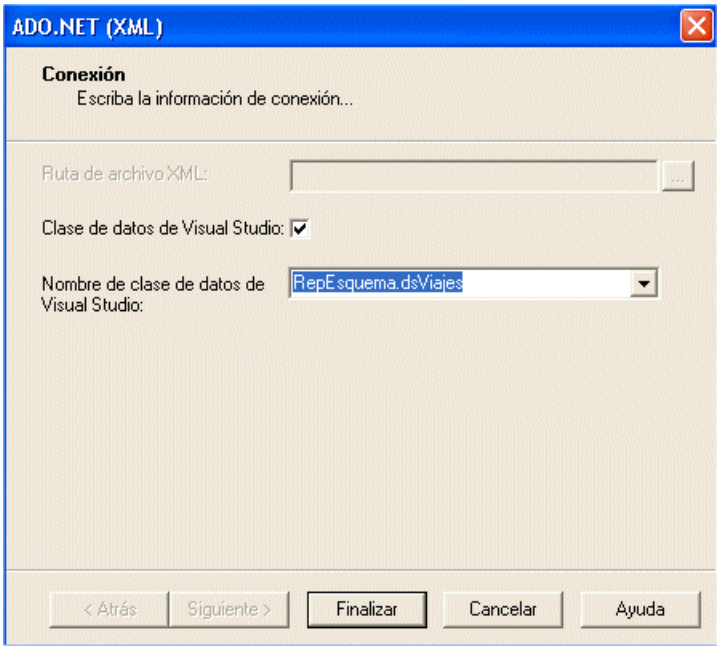


Figura 107. Conectando el informe con un DataSet creado manualmente.

Al volver al asistente de informes, el DataSet habrá sido reconocido por el entorno, pudiendo seleccionar las tablas (en este caso sólo una) que contiene, y agregarlas al informe, todo ello gracias a la magia de XML. Ver la Figura 108.

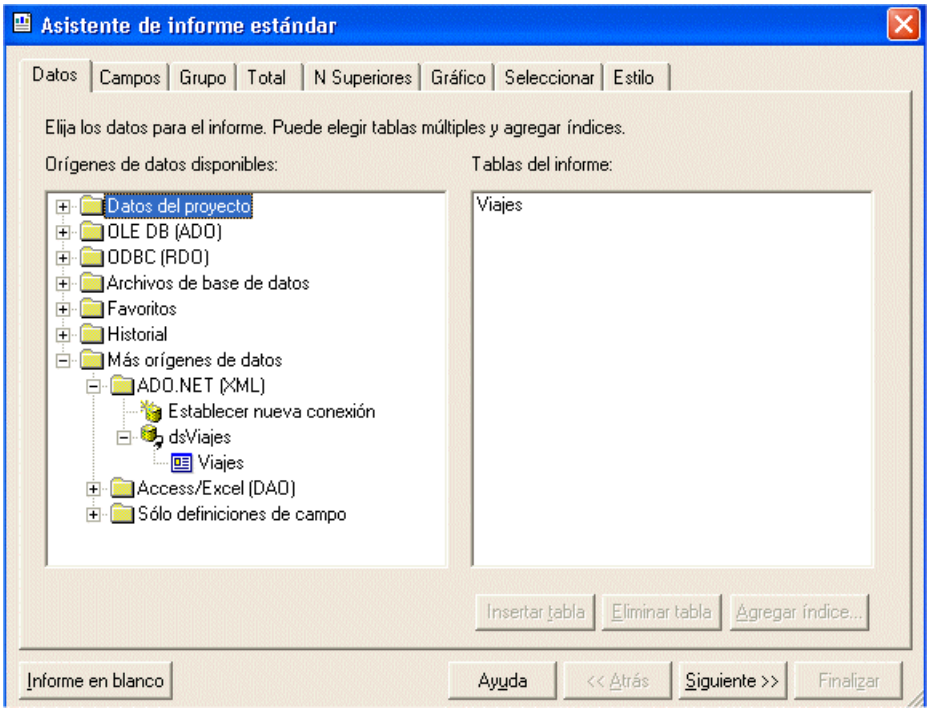


Figura 108. Agregar tablas del DataSet al informe.

Para el resto de pasos del asistente de informes, las operaciones a realizar son básicamente iguales que cuando hemos creado un informe a partir de una base de datos tradicional. Por ejemplo, la Figura 109 muestra cómo la selección de campos, una vez reconocida la estructura del DataSet, es igual que para una tabla normal.

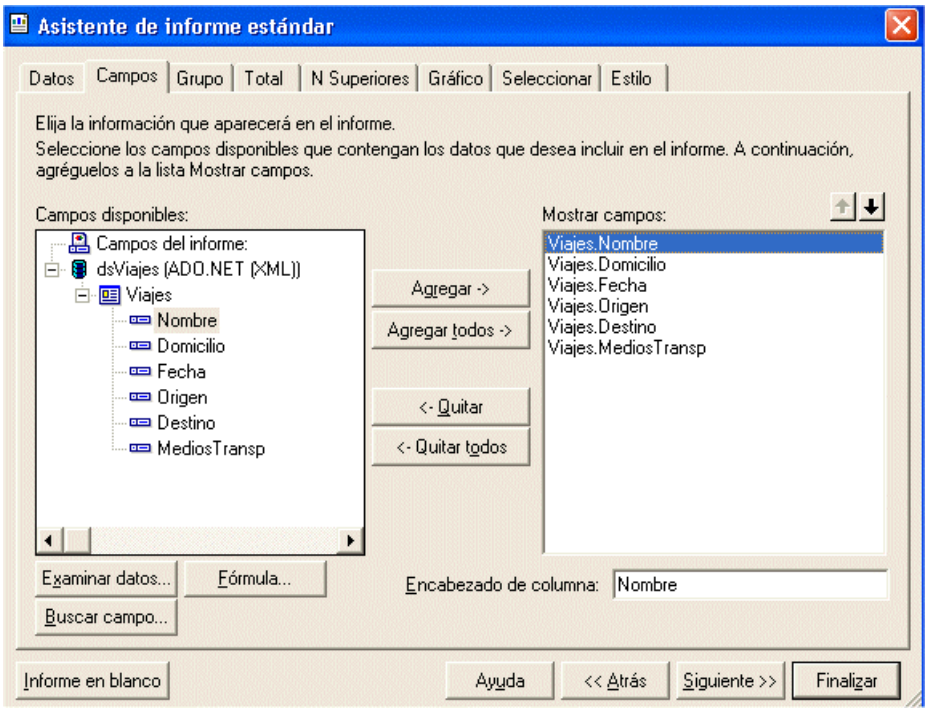


Figura 109. Selección de campos del DataSet para agregar al informe.

Tras finalizar el asistente de informes se generará el diseño del mismo, con la distribución de campos indicada, que tiene como origen el typed DataSet creado. Ciertamente sorprendente, como vemos en la Figura 110.

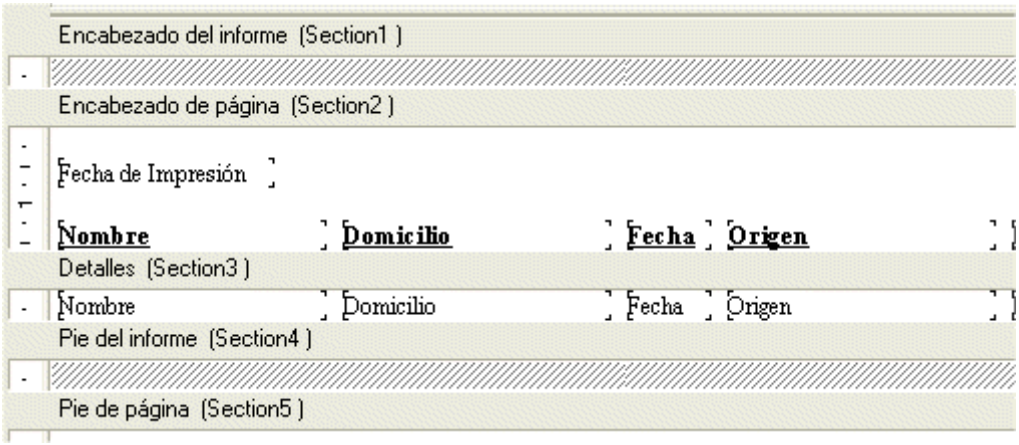


Figura 110. Diseño del informe creado a partir del DataSet manual.

A continuación crearemos un archivo XML con el nombre dsViajes.XML, que basado en el esquema antes creado, contenga los datos que serán posteriormente cargados en el informe, durante su ejecución. Para esta operación podemos utilizar un editor de texto plano o uno específico de XML; sin ir más lejos, el IDE de Visual Studio .NET permite la edición de este tipo de información. El Código fuente 25 muestra unos datos de prueba, aunque el lector puede escribir los que prefiera, siempre y cuando se atenga a la estructura del esquema.

```
<?xml version="1.0" standalone="yes" ?>
<dsViajes xmlns="http://www.tempuri.org/dsViajes.xsd">
	<Viajes>
		<Nombre>Julio Peral</Nombre>
		<Domicilio>Pza.Nueva, 7</Domicilio>
		<Fecha>07/10/2002</Fecha>
		<Origen>Pza.España</Origen>
		<Destino>Gran Vía</Destino>
		<MediosTransp>Metro L.3</MediosTransp>
	</Viajes>
	<Viajes>
		<Nombre>Raquel Sonseca</Nombre>
		<Domicilio>Carabineros, 12</Domicilio>
		<Fecha>15/09/2002</Fecha>
		<Origen>Goya</Origen>
		<Destino>Pza.Castilla</Destino>
		<MediosTransp>Metro L.4 - Bus L.75</MediosTransp>
	</Viajes>
	<Viajes>
		<Nombre>Ana Rimas</Nombre>
		<Domicilio>Duque de Montana, 50</Domicilio>
		<Fecha>25/10/2002</Fecha>
		<Origen>Ppe.Vergara</Origen>
		<Destino>Delicias</Destino>
		<MediosTransp>Metro L.6-3</MediosTransp>
	</Viajes>
</dsViajes>
```

Código fuente 25

Como paso final de todo este proceso, añadiremos al formulario del proyecto un control visualizador de informes, y en el evento Load( ) del formulario escribiremos las líneas del Código fuente 26, que realizarán la creación de los objetos y la carga de datos en el informe.

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    ' instanciar un objeto DataSet y un informe
    Dim dsDatosInforme As New DataSet()
    Dim oRptPrueba As New rptPrueba()

    ' leer con el DataSet el archivo xml
    dsDatosInforme.ReadXml("..\dsViajes.xml")
    ' cargar el informe con el DataSet
    oRptPrueba.SetDataSource(dsDatosInforme)
    ' mostrar el informe con el visualizador
    Me.crvInforme.ReportSource = oRptPrueba
End Sub
```

Código fuente 26

El resultado, que no deja de asombrarnos, gracias a las maravillas de XML, lo podemos ver en la Figura 111, en la que se muestra el informe con los datos obtenidos del archivo XML.

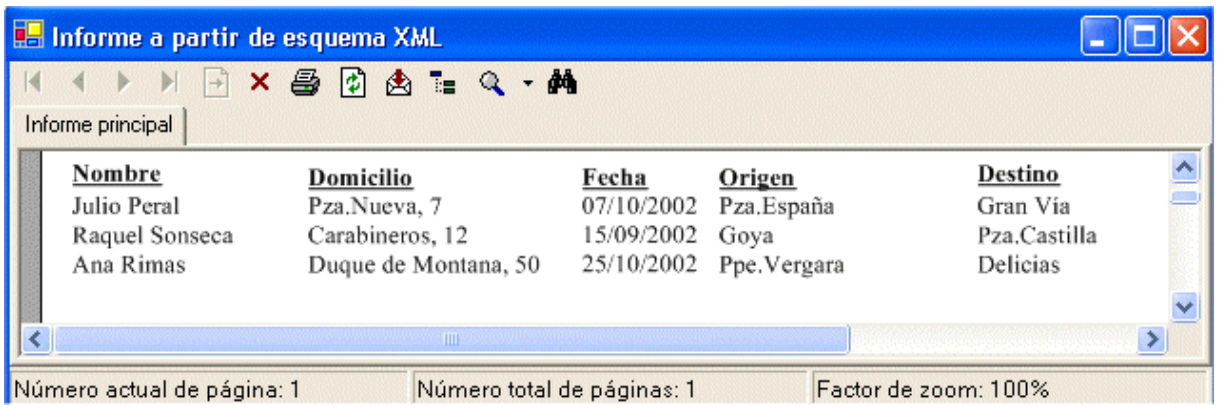


Figura 111. Informe mostrando datos de un archivo XML.

Exportar un informe a otros formatos de archivo

Una de las características más potentes de esta versión de Crystal Reports consiste en su capacidad para exportar un informe a un archivo que pueda ser consultado con una aplicación o utilidad externa.

De esta forma es posible generar, a partir de un informe, un archivo en alguno de los formatos especificados en la Tabla 1.

Tipo de archivo	Formato	Aplicación
PDF	Portable Document Format	Acrobat Reader
XLS	Hoja de cálculo Excel	Excel

DOC	Documento Word	Word
RTF	Rich Text Format	Word, Wordpad
HTM	Hyper Text Markup Language	Navegadores web: Internet Explorer, Navigator, etc.

Tabla 1. Formatos de exportación de Crystal Reports para Visual Studio .NET.

Como ejemplo ilustrativo de esta cualidad del motor de informes de CR, vamos a crear un proyecto con el nombre RepExportar (para acceder al proyecto de este ejemplo hacer clic aquí), que tenga un informe que muestre los registros de la tabla Categories.

A continuación vamos a diseñar el formulario del proyecto, incluyendo un ComboBox para seleccionar el tipo de formato, que contenga en su propiedad-colección Items los siguientes valores: PDF, Excel, Word, RTF y HTML; un TextBox para escribir el nombre del archivo al que vamos a exportar el informe; y un Button para realizar el proceso de exportación. El formulario quedaría como muestra la Figura 112.

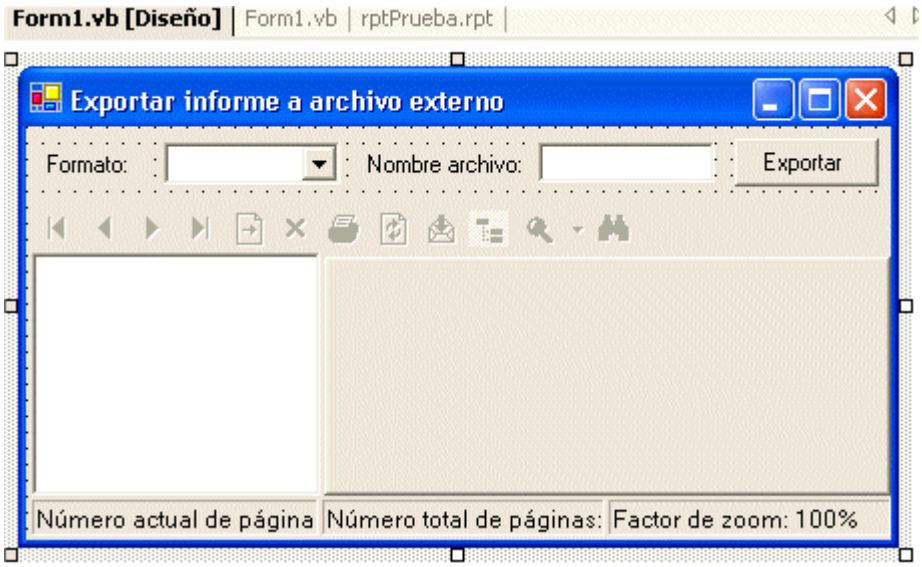


Figura 112. Formulario para exportar informe a otros formatos de archivo.

El funcionamiento del formulario será el siguiente: al comenzar la ejecución, durante el evento Load( ) del formulario, y al igual que en otros ejemplos, se visualizará el informe en el control CrystalReportViewer que contiene.

Posteriormente podremos elegir del ComboBox un tipo de formato de exportación, y escribir en el TextBox el nombre del archivo al que se volcará el informe. Al pulsar el botón del formulario, deberemos escribir el código mostrado en el Código fuente 27, para realizar el proceso de traspaso del listado a un archivo externo. El archivo generado será depositado en el directorio \bin, el mismo en el que es generado el ejecutable del proyecto.

```
Imports CrystalDecisions.Shared

Public Class Form1
```

```
   '....
    Private Sub btnExportar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnExportar.Click
        ' declarar:
        ' objeto informe
        Dim oRptPrueba As rptPrueba
        ' objeto para contener ruta y nombre de archivo
        ' destino de la exportación
        Dim oArchivoDestino As DiskFileDestinationOptions
        ' objeto para contener información adicional
        ' al exportar a un archivo HTML
        Dim oFormatoHTML As HTMLFormatOptions

        ' si no se especifica el formato y nombre de archivo, salir
        If Me.cboFormatos.Text.Length = 0 Then
            MessageBox.Show("Seleccionar un formato para exportar")
            Exit Sub
        End If

        If Me.txtArchivo.Text.Length = 0 Then
            MessageBox.Show("Escribir nombre de archivo para exportar")
            Exit Sub
        End If

        ' instanciar objetos
        oArchivoDestino = New DiskFileDestinationOptions()
        oRptPrueba = New rptPrueba()
        ' especificar que el destino de la exportación será un archivo en disco
        oRptPrueba.ExportOptions.ExportDestinationType =
ExportDestinationType.DiskFile

        ' en función del formato elegido,
        ' asignar al informe el tipo de formato en el que deberá
        ' exportar, y el nombre de archivo con la extensión adecuada
        ' al formato
        Select Case Me.cboFormatos.Text
            Case "PDF"
                oRptPrueba.ExportOptions.ExportFormatType =
ExportFormatType.PortableDocFormat
                oArchivoDestino.DiskFileName = Me.txtArchivo.Text & ".PDF"

            Case "Excel"
                oRptPrueba.ExportOptions.ExportFormatType = ExportFormatType.Excel
                oArchivoDestino.DiskFileName = Me.txtArchivo.Text & ".XLS"

            Case "Word"
                oRptPrueba.ExportOptions.ExportFormatType =
ExportFormatType.WordForWindows
                oArchivoDestino.DiskFileName = Me.txtArchivo.Text & ".DOC"

            Case "RTF"
                oRptPrueba.ExportOptions.ExportFormatType =
ExportFormatType.RichText
                oArchivoDestino.DiskFileName = Me.txtArchivo.Text & ".RTF"

            Case "HTML"
                ' en el caso de exportar a HTML, utilizar
                ' un objeto específico con información de
                ' formato para HTML
                oFormatoHTML = New HTMLFormatOptions()
                oFormatoHTML.HTMLFileName = Me.txtArchivo.Text
                ' asignar el objeto con el formato HTML
                ' a las opciones de formato del objeto informe
                oRptPrueba.ExportOptions.FormatOptions = oFormatoHTML

                oRptPrueba.ExportOptions.ExportFormatType = ExportFormatType.HTML40
                oArchivoDestino.DiskFileName = Me.txtArchivo.Text & ".HTM"
```

```
        End Select

        ' asignar la información del archivo a exportar al objeto informe
        oRptPrueba.ExportOptions.DestinationOptions = oArchivoDestino
        ' exportar el informe
        oRptPrueba.Export()

        MessageBox.Show("Informe exportado a " & Me.cboFormatos.Text)
    End Sub
End Class
```

Código fuente 27

Aunque se han incluido comentarios a lo largo del código, podemos añadir las siguientes explicaciones complementarias acerca de los objetos y miembros de los mismos que se utilizan en el ejemplo:

- **Export( )**. Método del objeto informe que realiza la acción de exportar el mismo a un archivo externo.

- **ExportOptions**. Clase que nos permite pasar diversa información al informe para realizar la exportación. En este ejemplo, ExportOptions es a su vez una propiedad del objeto informe, que contiene los siguientes miembros:

    - **ExportDestinationType**. Un tipo de la enumeración ExportDestinationType que indica el destino del informe. En nuestro caso un archivo en disco.

    - **ExportFormatType**. Un tipo de la enumeración ExportFormatType, que indica el formato de exportación: PDF, Excel, Word, etc.

    - **FormatOptions**. En el caso de exportar a HTML, utilizaremos esta propiedad para asignar al informe información específica acerca de este tipo de formato.

    - **DestinationOptions**. Un objeto de tipo DiskFileDestinationOptions, con la información del archivo al que se va a exportar el informe

- **DiskFileDestinationOptions**. Clase mediante la que especificamos, en su propiedad DiskFileName, una cadena con el nombre del archivo, extensión acorde al tipo de formato, y ruta opcional de disco, en el que vamos a depositarlo.

- **HTMLFormatOptions**. En el caso de que necesitemos exportar el listado a formato HTML, necesitaremos emplear también un objeto de esta clase, en el que con la propiedad HTMLFileName indicaremos el nombre del archivo a exportar. Dadas las características especiales de este formato, como habrá podido apreciar el lector, es el único caso en el ejemplo en el que tenemos que utilizar una clase adicional, para pasar la información del formato.

Finalizada la escritura del código, una vez que ejecutemos el ejemplo, podremos generar

En la Figura 113 y Figura 114, se muestra el informe de este ejemplo, pero visualizado desde los archivos a los que lo hemos exportado mediante nuestro proyecto.

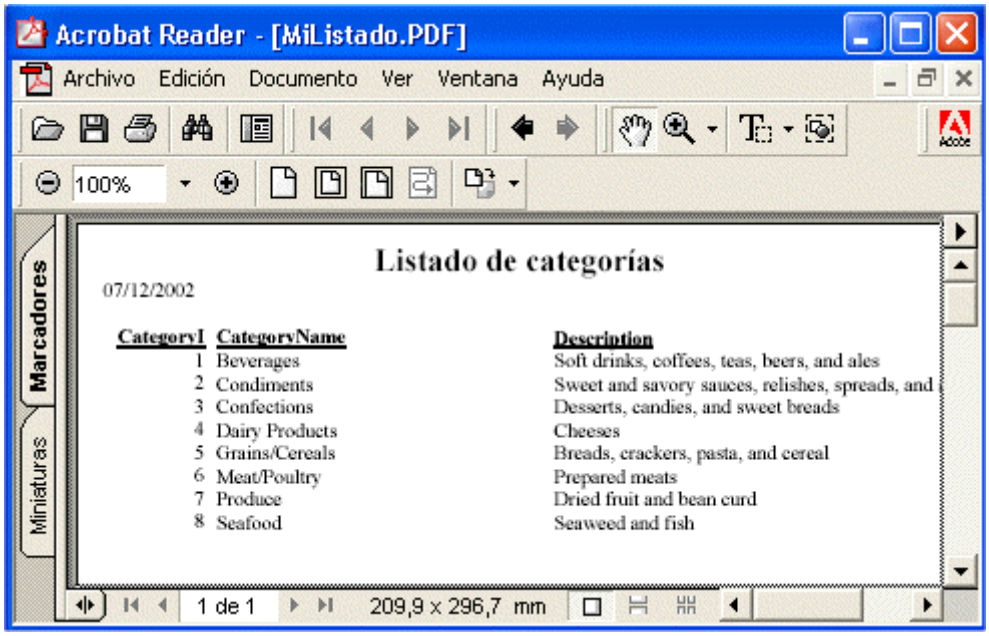


Figura 113. Informe exportado a PDF.

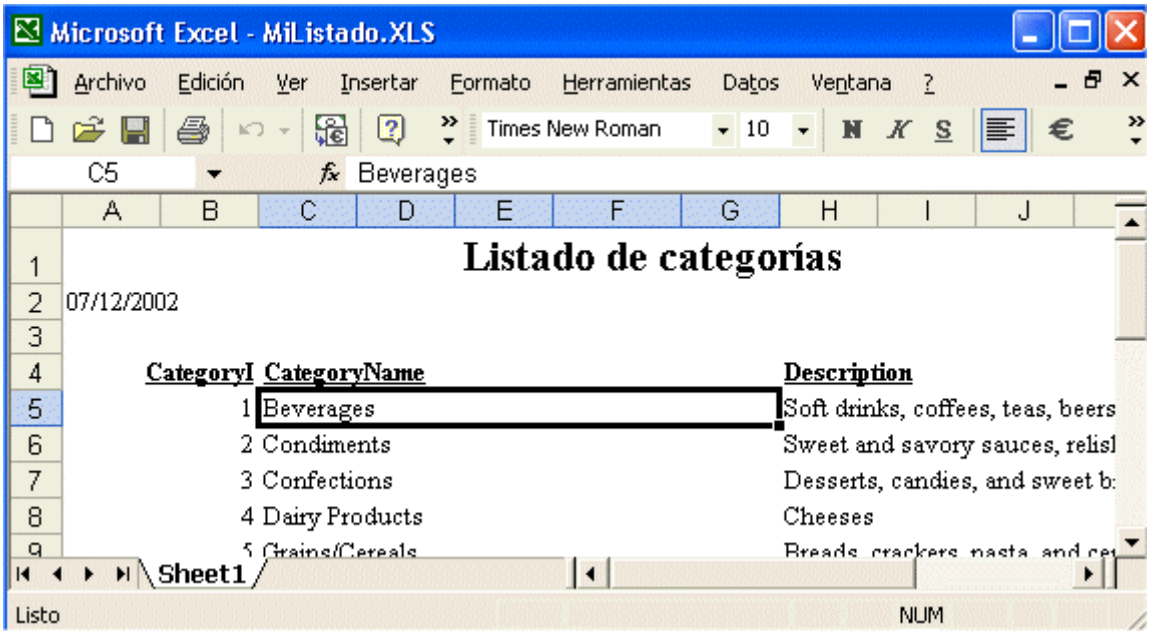


Figura 114. Informe exportado a Excel.

Manipulación por código del control CrystalReportViewer

Como ya hemos comprobado sobradamente durante los ejemplos realizados hasta el momento, el control CrystalReportViewer nos permite efectuar una previsualización de nuestro informe en pantalla, utilizando un formulario como base, evitando un innecesario consumo de tiempo y papel en las pruebas de ajuste del informe, previas a la obtención del diseño definitivo.

Este control aporta una barra de herramientas, con un conjunto de botones, para realizar diversas operaciones entre las que se encuentran las siguientes: navegación entre las páginas y desplazamiento

a una página concreta del informe; exportar a formatos externos; mostrar/ocultar grupos; ajuste del nivel de aumento o zoom; búsquedas de texto.

Todas estas acciones pueden llevarse a cabo también desde código, manipulando las propiedades y métodos de este control. Como demostración, vamos a crear un proyecto con el nombre RepCRVCodigo (para acceder al proyecto de este ejemplo hacer clic aquí), en el que añadiremos un informe que muestre la tabla Orders, con los registros agrupados por el campo ShipCountry.

El siguiente paso de este ejemplo consistirá en diseñar el formulario del proyecto, dotándole del aspecto que muestra la Figura 115, de manera que a través de controles Windows, y código asociado a los mismos, podamos realizar las mismas tareas que proporciona la barra de herramientas del control CrystalReportViewer.

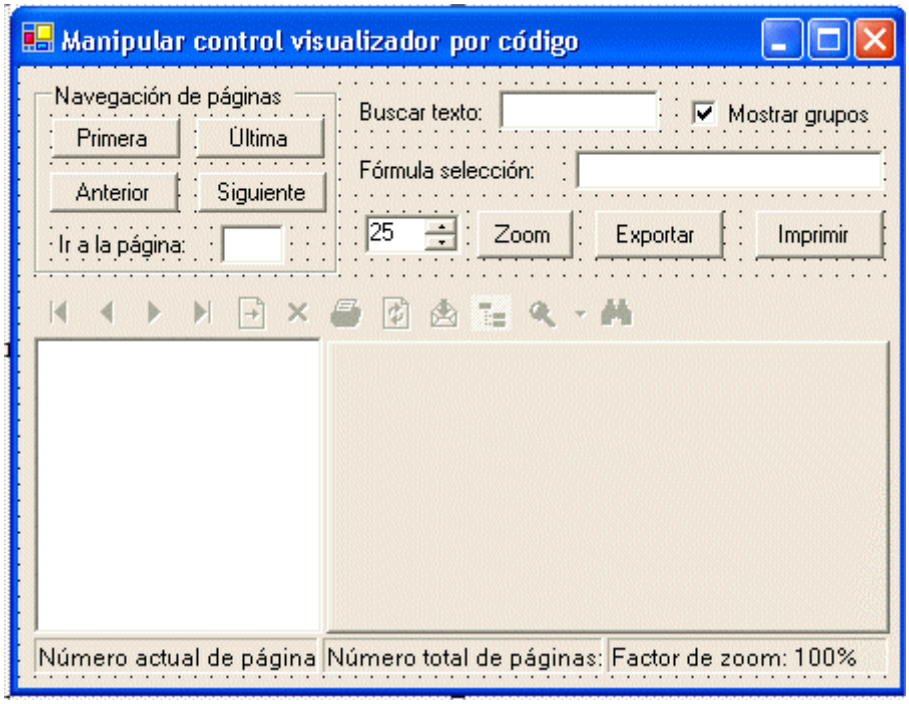


Figura 115. Formulario para manipular por código el control CrystalReportViewer.

Al asignar el informe al control visualizador durante la carga del formulario, ocultaremos la barra de herramientas del visualizador mediante su propiedad DisplayToolbar, por lo que las operaciones que hemos realizado habitualmente mediante los botones del control, las efectuaremos mediante los otros controles que hemos insertado en el formulario, y cuyo código se muestra al completo en el Código fuente 28, con los oportunos comentarios explicativos. Recomendamos al lector que ejecute el proyecto para ir probando cada una de las características que han sido implementadas por código.

```
Public Class Form1
    '....
    Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        ' instanciar objeto informe
        Dim oRptPrueba As New rptPrueba()
        ' ocultar barra herramientas control visualizador
        Me.crvInforme.DisplayToolbar = False
        ' cargar informe en control visualizador
        Me.crvInforme.ReportSource = oRptPrueba
```

```
    End Sub

    ' botones con operaciones de navegación por las páginas del informe
    Private Sub btnSiguiente_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSiguiente.Click
        Me.crvInforme.ShowNextPage()
    End Sub

    Private Sub btnAnterior_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnAnterior.Click
        Me.crvInforme.ShowPreviousPage()
    End Sub

    Private Sub btnPrimera_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnPrimera.Click
        Me.crvInforme.ShowFirstPage()
    End Sub

    Private Sub btnUltima_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnUltima.Click
        Me.crvInforme.ShowLastPage()
    End Sub

    Private Sub txtIrAPag_KeyPress(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles txtIrAPag.KeyPress
        ' al pulsar la tecla Intro...
        If e.KeyChar = ControlChars.Cr Then
            ' nos situamos en una página del informe
            If IsNumeric(Me.txtIrAPag.Text) Then
                Me.crvInforme.ShowNthPage(Me.txtIrAPag.Text)
            Else
                MessageBox.Show("El valor debe ser numérico")
                Me.txtIrAPag.Text = ""
            End If
        End If

    End Sub

    Private Sub btnImprimir_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnImprimir.Click
        ' enviar informe a la impresora,
        ' este método muestra la caja de diálogo del sistema
        ' para seleccionar impresora e imprimir
        Me.crvInforme.PrintReport()
    End Sub

    Private Sub btnExportar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnExportar.Click
        ' exportar informe a un archivo,
        ' este método muestra una caja de diálogo para
        ' seleccionar el tipo de formato a exportar, y
        ' dar el nombre al archivo
        Me.crvInforme.ExportReport()
    End Sub

    Private Sub chkGrupos_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles chkGrupos.CheckedChanged
        ' mostrar/ocultar panel de grupos del informe
        Me.crvInforme.DisplayGroupTree = Me.chkGrupos.Checked
    End Sub

    Private Sub txtBuscar_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles txtBuscar.KeyPress
        ' buscar una cadena de texto en el informe
        Me.crvInforme.SearchForText(Me.txtBuscar.Text)
    End Sub
```

```
    Private Sub btnZoom_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnZoom.Click
        ' aplicar nivel de aumento al visualizador del informe
        Me.crvInforme.Zoom(Me.udZoom.Value)
    End Sub

    Private Sub txtSeleccion_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles txtSeleccion.KeyPress
        ' al pulsar Intro en este control...
        If e.KeyChar = ControlChars.Cr Then
            ' asignar una fórmula de selección de registros al informe,
            ' por ejemplo: {Orders.EmployeeID} = 4
            Me.crvInforme.SelectionFormula = Me.txtSeleccion.Text
            ' actualizar el informe, para que se muestren
            ' los registros según la fórmula aplicada
            Me.crvInforme.RefreshReport()
        End If
    End Sub

End Class
```

Código fuente 28

Como nota aclarativa respecto a la aplicación de una fórmula de selección de registros, hemos de indicar que en el TextBox habilitado a tal efecto, tenemos que escribir la expresión de la fórmula, tal y como lo haríamos en el código fuente, a excepción de las comillas delimitadoras de la cadena que contiene la fórmula. La Figura 116 muestra un ejemplo de este particular.

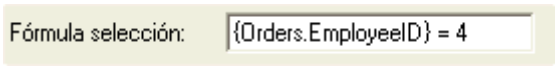


Figura 116. Escritura de la fórmula de selección de registros desde el formulario.

Creación de informes en ASP.NET

La elaboración de informes de Crystal Reports, que deban ser visualizados en un formulario Web de un proyecto ASP.NET, salvando ciertas peculiaridades propias de la naturaleza de ejecución en Internet, es un proceso muy parecido al realizado para aplicaciones basadas en formularios Windows.

Como ejemplo, vamos a crear un nuevo proyecto, esta vez de tipo Aplicación Web ASP.NET, con el nombre RepWeb (para acceder al proyecto de este ejemplo hacer clic aquí), que contenga un informe que muestre los registros de la tabla Customers, agrupados por el campo Country. La fase de creación del informe es igual que para una aplicación Windows, que ya conocemos de anteriores apartados.

A continuación pasaremos al diseñador del formulario Web e insertaremos un control CrystalReportViewer, cuya misión es visualizar el informe aunque con las características particulares de una aplicación Web. Este control, una vez dibujado en el formulario, muestra la apariencia de la Figura 117.

Por último, sólo quedaría escribir el código en el evento Load( ) del WebForm por ejemplo, que cargara el informe en el control visualizador. Ver el Código fuente 29.

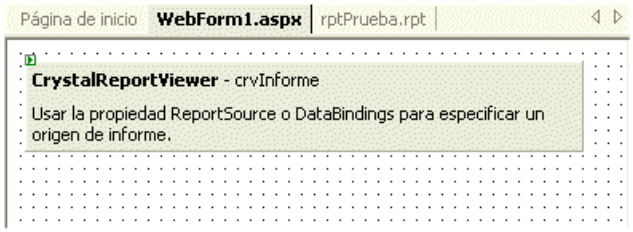


Figura 117. Control CrystalReportViewer en un formulario Web.

```
Public Class WebForm1
    Inherits System.Web.UI.Page
    '....

    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim oRptPrueba As New rptPrueba() ' instanciar el objeto informe
        Me.crvInforme.ReportSource = oRptPrueba ' asignar el informe al control
    End Sub

End Class
```

Código fuente 29

Como puede comprobar el lector, las líneas a emplear son iguales que para un informe de una aplicación Windows.

En este caso es importante tener en cuenta que para cada modificación que efectuemos, debemos volver a generar el proyecto con la opción de menú del IDE *Generar + Volver a generar <NombreProyecto>*, ya que en caso contrario, cuando ejecutemos la aplicación en el navegador, las modificaciones realizadas pudieran no ser reflejadas.

La Figura 118 muestra esta aplicación en ejecución, con el informe cargado en el formulario Web del navegador.

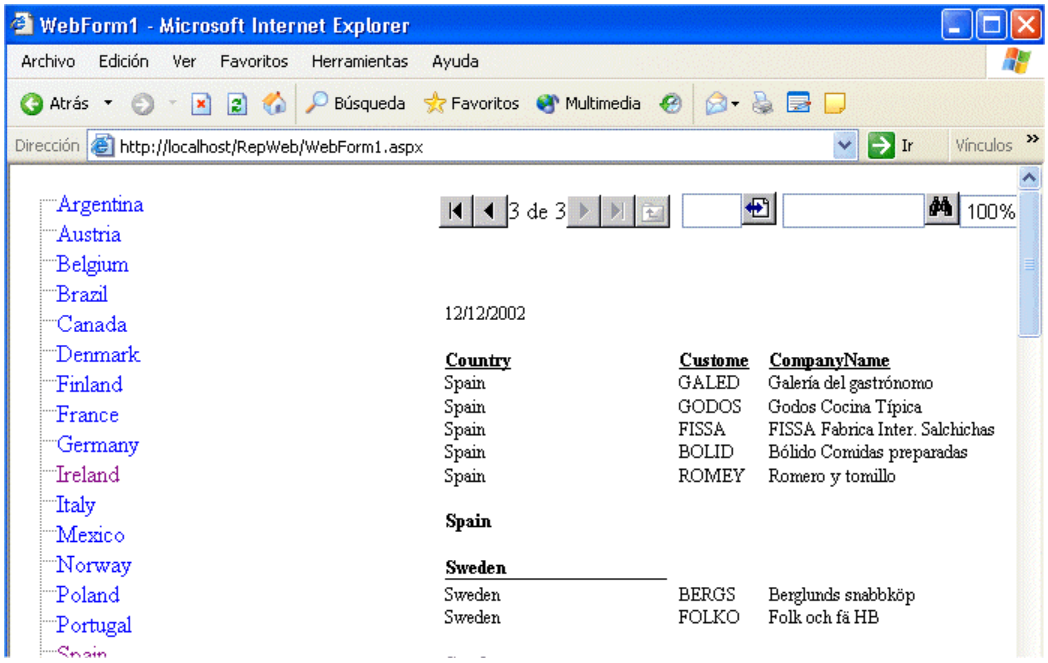


Figura 118. Informe mostrado en una aplicación ASP.NET.

Informes gestionados en memoria caché

Para conseguir un mayor rendimiento, y optimizar la ejecución de los informes de Crystal Reports .NET en una aplicación ASP.NET, el motor de CR genera internamente, para cada informe diseñado en un proyecto, un archivo con clases que se asocian al informe creado desde el diseñador.

Este aspecto fue tratado de forma introductoria en el apartado *La clase del informe*. Ahora vamos a profundizar un poco más en la arquitectura diseñada por CR para los informes ejecutados en el entorno Web.

Una de las misiones de las clases generadas para el informe, consiste en mantener en caché el informe en ejecución, minimizando de esta manera el número de *viajes* que han de ser realizados al servidor para obtener los datos, si dichos datos no han sido modificados, aspecto este que supone un importante ahorro de recursos y tiempo en la ejecución.

Suponiendo que hemos creado un informe con el nombre rptPrueba, y seleccionando el menú *Proyecto + Mostrar todos los archivos*, accederemos al archivo rptPrueba.VB, que contiene la clase rptPrueba, que hereda de ReportClass, y se trata de la clase que permite la creación de informes strong typed, o con establecimiento inflexible de tipos. Esta clase da el acceso a las secciones del informe a través de sus miembros, actuando como una clase envoltorio o wrapper. Veamos parte de su contenido en el Código fuente 30.

```
Option Strict Off
Option Explicit On


Imports CrystalDecisions.CrystalReports.Engine
Imports CrystalDecisions.ReportSource
Imports CrystalDecisions.Shared
Imports System
Imports System.ComponentModel


Public Class rptPrueba
    Inherits ReportClass


    Public Sub New()
        MyBase.New
    End Sub


    Public Overrides Property ResourceName As [String]
        Get
            Return "rptPrueba.rpt"
        End Get
        Set
            'Do nothing
        End Set
    End Property

    <Browsable(false), _
     DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)> _
    Public ReadOnly Property Section1 As Section
        Get
            Return Me.ReportDefinition.Sections(0)
        End Get
    End Property
```

```
    '....

End Class
```

Código fuente 30

El mismo archivo rptPrueba.VB, contiene también la clase CachedrptPrueba, que implementa el interfaz ICachedReport, y como estos nombres nos indican, permite la creación de informes mantenidos en memoria caché o intermedia. El Código fuente 31 muestra parte del código de esta clase.

```
<System.Drawing.ToolboxBitmapAttribute(GetType(CrystalDecisions.Shared.ExportOption
s), "report.bmp")>  _
Public Class CachedrptPrueba
    Inherits Component
    Implements ICachedReport

    Public Sub New()
        MyBase.New
    End Sub

    Public Overridable Property IsCacheable As [Boolean] Implements
CrystalDecisions.ReportSource.ICachedReport.IsCacheable
        Get
            Return true
        End Get
        Set
            '
        End Set
    End Property

    '....

End Class
```

Código fuente 31

Cuando se crea un informe en caché desde una aplicación ASP.NET, por ejemplo, esta clase CachedrptPrueba actúa también a su vez como clase envoltorio de la clase correspondiente al informe strong typed rptPrueba.

Como ejemplo demostrativo del funcionamiento de informes en caché, vamos a crear una aplicación ASP.NET con el nombre RepCacheInforme (para acceder al proyecto de este ejemplo hacer clic aquí), en la que incluiremos un informe con el nombre rptPrueba, que muestre la tabla Products, y en la sección de encabezado de página añadiremos desde el Explorador de campos, un campo *Hora de los datos*, que se encuentra en el nodo *Campos especiales*.

Tras añadir un control CrystalReportViewer al formulario Web del proyecto, insertaremos desde el Cuadro de herramientas, en la pestaña Componentes, un objeto ReportDocument, mostrándose el cuadro de diálogo de la Figura 119.

En este caso, como al ejecutar el informe queremos que sea mantenido en caché, dejaremos marcado el CheckBox *Generar informe con tipos declarados en caché* de esta ventana, y la aceptaremos.

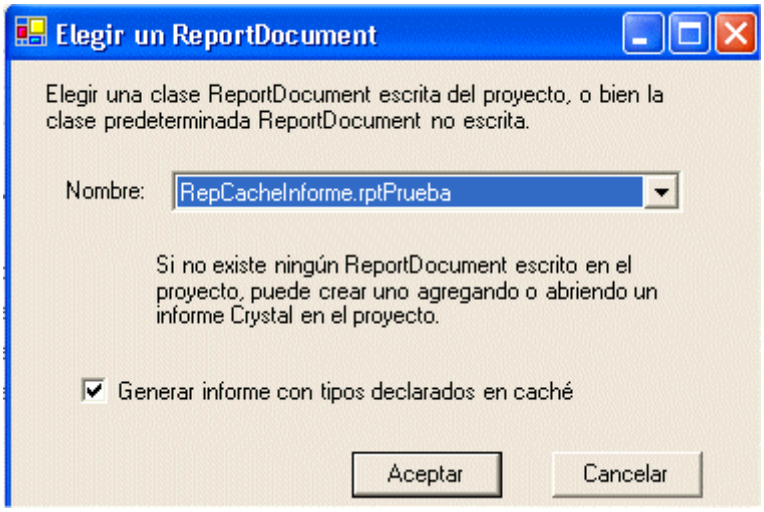


Figura 119. Creación de un objeto ReportDocument para un informe en caché en un formulario Web.

La anterior operación, creará en el panel de elementos ocultos del formulario Web un nuevo objeto ReportDocument con el nombre cachedrptPrueba1, que será el encargado de manipular el informe en ejecución.

A continuación escribiremos en el evento de carga del formulario Web el código que toma el objeto ReportDocument que contiene el informe y lo asigna al control visualizador. Ver el Código fuente 32.

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    Me.crvInforme.ReportSource = Me.cachedrptPrueba1
End Sub
```

Código fuente 32

Finalmente generaremos el proyecto, y abriremos Internet Explorer o el navegador web que utilicemos, y ejecutaremos nuestra aplicación ASP.NET. El informe que se muestra en el navegador queda en caché, observe el lector que aunque recarguemos la página, el campo de hora de obtención de los datos permanece invariable, signo indicativo de que por cada recarga de la página no se está realizando un viaje al servidor de datos para obtener de nuevo la tabla. Esto significa que el motor de informes tomó los datos en la primera carga, y ya los mantiene en las posteriores. Ver la Figura 120.

Este comportamiento, sin embargo, no será el más idóneo en determinadas ocasiones, ya que nos puede interesar, que cada vez que carguemos la página, se obtengan nuevamente los datos, por si se hubieran producido modificaciones. Para ello vamos a insertar en nuestro formulario Web un nuevo objeto ReportDocument, pero en este caso es muy importante que al aparecer la caja de diálogo de selección del informe de este objeto, desmarquemos el CheckBox *Generar informe con tipos declarados en caché*. Como resultado obtendremos en el panel de elementos ocultos del formulario un objeto ReportDocument con el nombre rptPrueba2.

En el evento de carga del formulario Web, comentaremos la línea de código que había previamente, y cargaremos ahora el control visualizador utilizando este último objeto rptPrueba2. Ver el Código fuente 33.

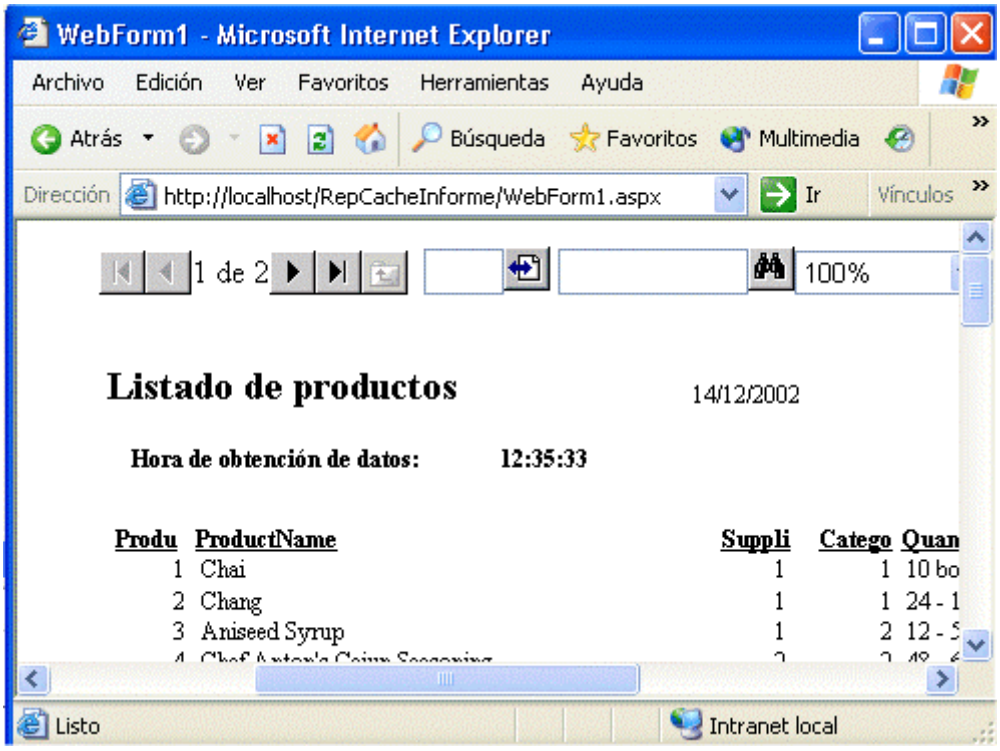


Figura 120. Ejecución de informe mantenido en caché.

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    ' deshabilitado --> Me.crvInforme.ReportSource = Me.cachedrptPrueba1
    Me.crvInforme.ReportSource = Me.rptPrueba2
End Sub
```

Código fuente 33

A continuación es muy importante volver a generar el proyecto, para que cuando lo ejecutemos en el navegador de Internet se utilice la versión con los últimos cambios realizados. A partir de ahora, al ejecutar esta aplicación en el navegador, cada vez que recarguemos la página ASP.NET, o al desplazarnos por las páginas del informe, el campo que muestra la hora de obtención de los datos será actualizado, indicándonos que el motor de CR ha tenido que ir al servidor de datos y recargar el informe con los registros que hubiera en ese momento en la tabla.

Actualización de la versión del informe mantenido en caché

En el ejemplo realizado en el apartado anterior existe una situación que no hemos tratado, ¿qué ocurre si queremos mantener el informe en caché, pero que también disponga de la capacidad de ser actualizado en determinados momentos, de modo que refleje los cambios que se producen en la fuente de datos a la que está conectado?

La respuesta a este problema la hallamos en la clase caché del informe, si seguimos con el proyecto de ejemplo comenzado en el apartado anterior, esta clase sería CachedrptPrueba.

Cada vez que se carga el informe en la página ASP.NET, se realizan llamadas al método GetCustomizedCacheKey( ) contenido en esta clase, cuyo objetivo es devolver un valor o clave de tipo cadena. Cuando dicha cadena cambia con respecto a la llamada anterior, el motor de CR asume que se quiere volver a cargar el informe desde su origen, descartando el que hay en caché.

Dentro del método GetCustomizedCacheKey( ), hay unas líneas comentadas, que ejecutan el método RequestContest.BuildCompleteCacheKey( ), que mediante un algoritmo, es el que se encarga de generar la clave que se devuelve como indicador de que el informe en caché sea o no válido. En nuestro caso no sería necesario, ya que lo que vamos a hacer es que cada vez que sea llamado GetCustomizedCacheKey( ), devolveremos de la hora actual, la parte correspondiente a los minutos, de forma que cuando estos cambien, se descartará el informe que hay en caché y se volverá a generar uno nuevo, depositándolo en caché y sustituyendo al que había previamente. Veamos los cambios que hemos introducido en el Código fuente 34.

```
<System.Drawing.ToolboxBitmapAttribute(GetType(CrystalDecisions.Shared.ExportOption
s), "report.bmp")>  _
Public Class CachedrptPrueba
    Inherits Component
    Implements ICachedReport

    '....

    Public Overridable Function GetCustomizedCacheKey(ByVal request As
RequestContext) As [String] Implements
CrystalDecisions.ReportSource.ICachedReport.GetCustomizedCacheKey
        Dim key As [String] = Nothing
        '// The following is the code used to generate the default
        '// cache key for caching report jobs in the ASP.NET Cache.
        '// Feel free to modify this code to suit your needs.
        '// Returning key == null causes the default cache key to
        '// be generated.
        '
        'key = RequestContext.BuildCompleteCacheKey(
        '    request,
        '    null,       // sReportFilename
        '    this.GetType(),
        '    this.ShareDBLogonInfo );

        ' cada vez que este método sea llamado
        ' obtendremos la hora actual, y devolveremos
        ' la parte correspondiente a los minutos;
        ' cuando los minutos cambien, se volverá a generar
        ' el informe, sustituyendo el que había hasta ese
        ' momento en caché
        Dim HoraActual As New DateTime()
        HoraActual = DateTime.Now
        key = HoraActual.Minute
        Return key
    End Function
End Class
```

Código fuente 34

Seguidamente volveremos en el evento Page_Load( ) de la página ASP.NET, a dejar el objeto ReportDocument que utilizaba el informe con caché. Ver Código fuente 35.

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
```

```
    Me.crvInforme.ReportSource = Me.cachedrptPrueba1

End Sub
```

Código fuente 35

A continuación generaremos de nuevo el proyecto y lo ejecutaremos desde el navegador. En este caso, cada vez que cambien los minutos, observe el lector, cómo la hora que hemos incluido también se actualiza, indicándonos que se ha producido una nueva generación del informe, trasladándose esta nueva copia a caché.

Esta técnica tiene, no obstante, un importante problema, si abrimos el diseñador del informe y modificamos algún elemento del mismo, se generará de nuevo la clase asociada al informe, perdiendo el código que hemos añadido.

Para evitar este inconveniente, una solución pasa por la escritura de nuestra propia clase que tenga el mismo código que la clase CachedrptPrueba más el código personalizado que necesitaramos nosotros añadir.

Vamos por lo tanto a crear una nueva clase con el nombre rptCacheProductos con el mencionado código. Ver el Código fuente 36.

```
Imports CrystalDecisions.CrystalReports.Engine
Imports CrystalDecisions.ReportSource
Imports CrystalDecisions.Shared
Imports System
Imports System.ComponentModel

Public Class rptCacheProductos
    Inherits Component
    Implements ICachedReport

    Public Sub New()
        MyBase.New()
    End Sub

    Public Overridable Property IsCacheable() As [Boolean] Implements
CrystalDecisions.ReportSource.ICachedReport.IsCacheable
        Get
            Return True
        End Get
        Set(ByVal Value As [Boolean])
            '
        End Set
    End Property

    Public Overridable Property ShareDBLogonInfo() As [Boolean] Implements
CrystalDecisions.ReportSource.ICachedReport.ShareDBLogonInfo
        Get
            Return False
        End Get
        Set(ByVal Value As [Boolean])
            '
        End Set
    End Property

    Public Overridable Property CacheTimeOut() As TimeSpan Implements
CrystalDecisions.ReportSource.ICachedReport.CacheTimeOut
        Get
```

```
            Return CachedReportConstants.DEFAULT_TIMEOUT
        End Get
        Set(ByVal Value As TimeSpan)
            '
        End Set
    End Property

    Public Overridable Function CreateReport() As ReportDocument Implements
CrystalDecisions.ReportSource.ICachedReport.CreateReport
        Dim rpt As rptPrueba = New rptPrueba()
        rpt.Site = Me.Site
        Return rpt
    End Function

    Public Overridable Function GetCustomizedCacheKey(ByVal request As
RequestContext) As [String] Implements
CrystalDecisions.ReportSource.ICachedReport.GetCustomizedCacheKey
        Dim key As [String] = Nothing
        '// The following is the code used to generate the default
        '// cache key for caching report jobs in the ASP.NET Cache.
        '// Feel free to modify this code to suit your needs.
        '// Returning key == null causes the default cache key to
        '// be generated.
        '
        'key = RequestContext.BuildCompleteCacheKey(
        '    request,
        '    null,       // sReportFilename
        '    this.GetType(),
        '    this.ShareDBLogonInfo );

        ' cada vez que este método sea llamado
        ' obtendremos la hora actual, y devolveremos
        ' la parte correspondiente a los minutos;
        ' cuando los minutos cambien, se volverá a generar
        ' el informe, sustituyendo el que había hasta ese
        ' momento en caché
        Dim HoraActual As New DateTime()
        HoraActual = DateTime.Now
        key = HoraActual.Minute

        Return key
    End Function
End Class
```

Código fuente 36

Esta clase, al no estar ligada a ningún diseñador de informes, no se verá afectada por los cambios que podamos hacer en nuestro informe.

Como último toque para dejar nuestro ejemplo adecuadamente operativo, volveremos de nuevo al evento de carga de la página ASP.NET e instanciaremos un objeto de esta clase que acabamos de crear, y será dicho objeto el que pasemos al control visualizador del informe, como muestra el Código fuente 37.

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    ' instanciar un objeto de nuestra clase propia
    ' de creación de informes en caché
    Dim oRptCacheProductos As New rptCacheProductos()
    ' asignar el objeto al visualizador del informe
    Me.crvInforme.ReportSource = oRptCacheProductos
```

```
End Sub
```

Código fuente 37

Publicación de un informe como un Web Service

Otra de las novedades a resaltar en Crystal Reports para Visual Studio .NET es su capacidad de poder publicar un informe como un servicio Web, con las ventajas que esta nueva tecnología de la plataforma .NET conlleva. Si a esta facultad unimos la sencillez con que un informe se puede publicar como un Web Service, la potencia de nuestros desarrollos se verá mejorada en muchos puntos.

A modo de ejemplo, vamos a crear un nuevo proyecto de tipo *Servicio Web ASP.NET*, al que daremos el nombre RepInformeWS (para acceder al proyecto de este ejemplo hacer clic aquí). En primer lugar, eliminaremos el servicio Web Service1.asmx que por defecto contiene, añadiendo un informe con el nombre rptSuppliers, que muestre los registros de la tabla Suppliers.

Una vez creado el informe, publicarlo como un servicio Web es algo tan sencillo como ir a la ventana Explorador de soluciones, hacer clic derecho sobre el informe, y seleccionar del menú contextual la opción *Publicar como servicio Web*, como muestra la Figura 121.

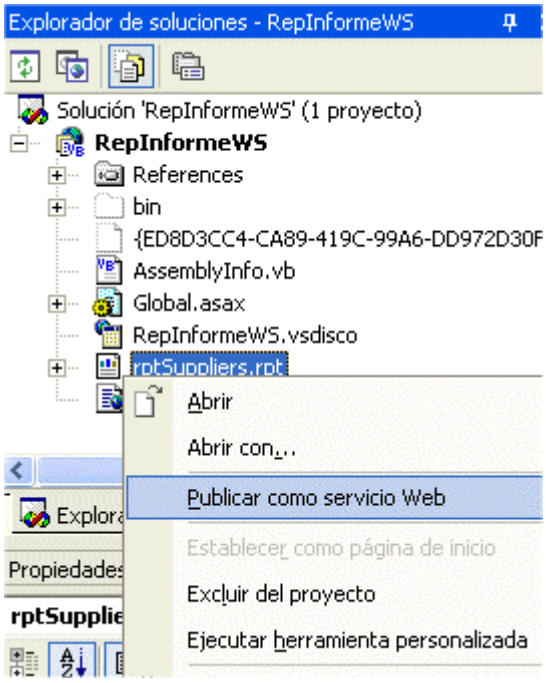


Figura 121. Publicar un informe como un servicio Web.

Una vez publicado el informe como un servicio Web, se añadirá al proyecto un nuevo elemento con el mismo nombre del informe, terminado en Service; en nuestro caso, en el Explorador de soluciones aparecerá este servicio Web con el nombre rptSuppliersService.asmx.

La técnica de ejecución del servicio Web es similar a la empleada para los informes ejecutados en páginas ASP.NET, ya que se utiliza un sistema de caché para almacenar el informe en memoria intermedia.

Si pasamos al Explorador de soluciones, en la vista de esta ventana que muestra todos los archivos, observaremos como asociado al informe/servicio Web, hay un archivo de código con el nombre rptSuppliersService.asmx.vb. Editando este archivo, veremos que contiene dos clases: rptSuppliersService, que hereda de ReportServiceBase, para proporcionar la funcionalidad de servicio Web; y por otro lado encontramos, anidada a esta clase, otra clase con el nombre CachedWebrptSuppliers, que implementa el interfaz ICachedReport, siendo su cometido el mantenimiento en caché del informe, como vimos en el ejemplo anterior, pero ahora bajo el esquema de funcionamiento de los servicios Web. El contenido de este archivo se muestra en el Código fuente 38.

```
Imports System
Imports System.Web.Services
Imports CrystalDecisions.Shared
Imports CrystalDecisions.CrystalReports.Engine
Imports CrystalDecisions.ReportSource
Imports CrystalDecisions.Web.Services


< WebService( Namespace:="http://crystaldecisions.com/reportwebservice/9.1/" ) > _
Public Class rptSuppliersService
    Inherits ReportServiceBase

    Public Sub New()
        Me.ReportSource = New CachedWebrptSuppliers(Me)
    End Sub

    Protected Sub OnInitReport(ByVal source As Object, ByVal args As EventArgs)
    End Sub


    Public Class CachedWebrptSuppliers
        Implements ICachedReport

        Protected webService As rptSuppliersService


        Public Sub New(ByVal webServiceParam As rptSuppliersService)
            Me.webService = webServiceParam
        End Sub

        Public Overridable Property IsCacheable() As Boolean _
                Implements ICachedReport.IsCacheable
            Get
                Return True
            End Get
            Set(ByVal Value As Boolean)
            End Set
        End Property

        Public Overridable Property ShareDBLogonInfo() As Boolean _
                Implements ICachedReport.ShareDBLogonInfo
            Get
                Return False
            End Get
            Set(ByVal Value As Boolean)
            End Set
        End Property

        Public Overridable Property CacheTimeOut() As TimeSpan _
                Implements ICachedReport.CacheTimeOut
            Get
                Return (CachedReportConstants.DEFAULT_TIMEOUT)
            End Get
```

```
            Set(ByVal Value As TimeSpan)
            End Set
        End Property

        Protected Overridable Function CreateReport() As ReportDocument _
                Implements ICachedReport.CreateReport
            Dim report As rptSuppliers = New rptSuppliers()

            AddHandler report.InitReport, New EventHandler(AddressOf
Me.webService.OnInitReport)

            Return report
        End Function

        Public Overridable Function GetCustomizedCacheKey(ByVal request As
RequestContext) As String _
                Implements ICachedReport.GetCustomizedCacheKey
            Dim key As String = Nothing

            '    key = RequestContext.BuildCompleteCacheKey(Nothing, Nothing,
Me.GetType(), Me.ShareDBLogonInfo)

            Return key
        End Function
    End Class 'CachedWebrptSuppliers

End Class
```

Código fuente 38

En lo que atañe al proyecto que contiene el informe publicado como Web Service, sólo nos quedaría generar el proyecto y ya estaría listo. Como habrá podido comprobar el lector ha sido una tarea muy sencilla.

Respecto a la parte cliente que haga uso del informe creado como Web Service, las tareas a realizar serían también muy fáciles. Vamos a crear un proyecto de tipo Windows, con el nombre ClienteRepWS (para acceder al proyecto de este ejemplo hacer clic aquí).

Pasando al Explorador de soluciones y haciendo clic derecho en el elemento Referencias, seleccionaremos la opción del menú contextual *Agregar referencia web*, que abrirá la ventana de búsqueda de servicios Web, en la que introduciremos la ruta de nuestro servicio, tal y como muestra la Figura 122.

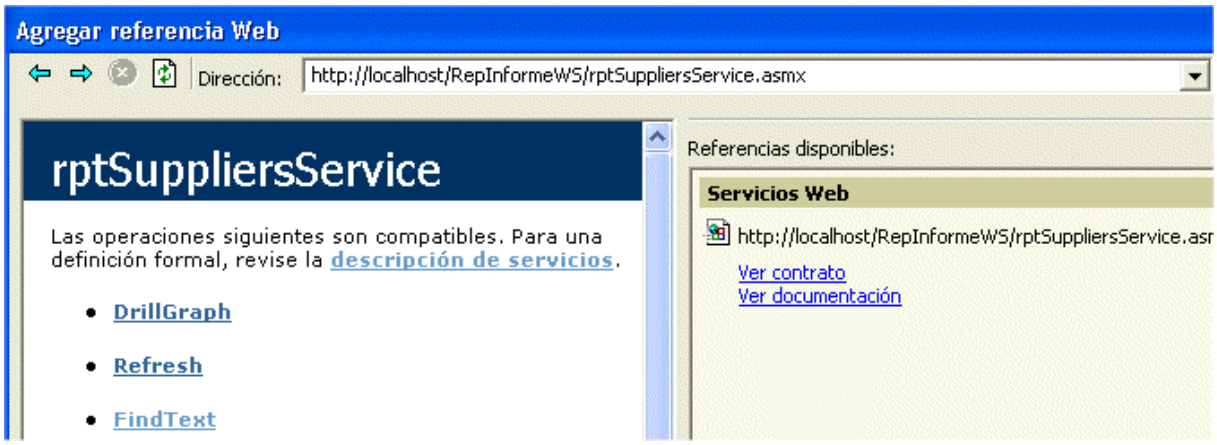


Figura 122. Utilidad para agregar referencia a un servicio Web en un proyecto cliente.

Al pulsar el botón *Agregar referencia* de esta ventana, se añadirá la referencia al Web Service del informe en nuestra aplicación cliente, de forma que ya podremos utilizarlo desde este proyecto. La Figura 123 muestra como quedaría el Explorador de soluciones después de esta operación.

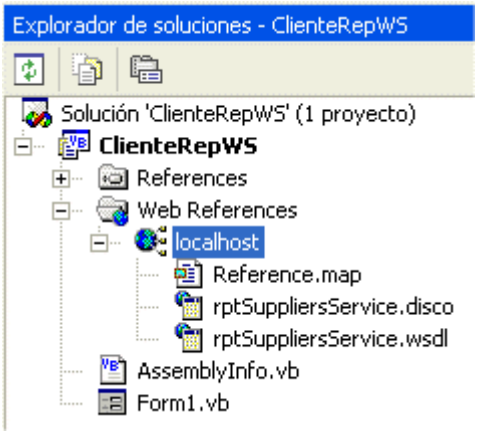


Figura 123. Explorador de soluciones conteniendo la referencia al servicio Web del informe.

Para invocar ahora al servicio Web podemos hacerlo de dos modos, que se muestran en el evento Load( ) del formulario, en el Código fuente 39.

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    ' ocultar el panel de grupos del control visualizador
    Me.crvInforme.DisplayGroupTree = False

    ' modos de invocar al servicio Web
    ' que contiene el informe:

    ' 1)
    ' asignar al control visualizador una cadena
    ' con la ruta del servicio Web
    'Me.crvInforme.ReportSource =
"http://localhost/RepInformeWS/rptSuppliersService.asmx"

    ' 2)
    ' instanciar un objeto del servicio Web informe,
    ' y asignarlo al control visualizador
    Dim oInformeWebServ As localhost.rptSuppliersService
    oInformeWebServ = New localhost.rptSuppliersService()
    Me.crvInforme.ReportSource = oInformeWebServ

End Sub
```

Código fuente 39

La ejecución de un informe publicado como un Web Service, no es impedimento para que durante el transcurso del programa podamos interactuar con él. Por ejemplo, vamos a añadir algunos controles adicionales al formulario que nos permitan, al pulsar un botón, asignar una fórmula de selección de registros al informe, y refrescar la vista que hasta ese momento tenemos en el formulario del informe. El Código fuente 40 muestra el código del botón que ejecuta las acciones necesarias.

```
Private Sub btnActualizar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnActualizar.Click
    ' construir la fórmula de selección de registros
    ' para el informe y asignarla al control visualizador
    Me.crvInforme.SelectionFormula = "{Suppliers.Country} = '" & Me.txtPais.Text &
"'"
    ' actualizar la vista del informe para que
    ' refleje la fórmula que acabamos de asignar
    Me.crvInforme.RefreshReport()
End Sub
```

Código fuente 40

Una muestra del resultado la podemos ver en la Figura 124.

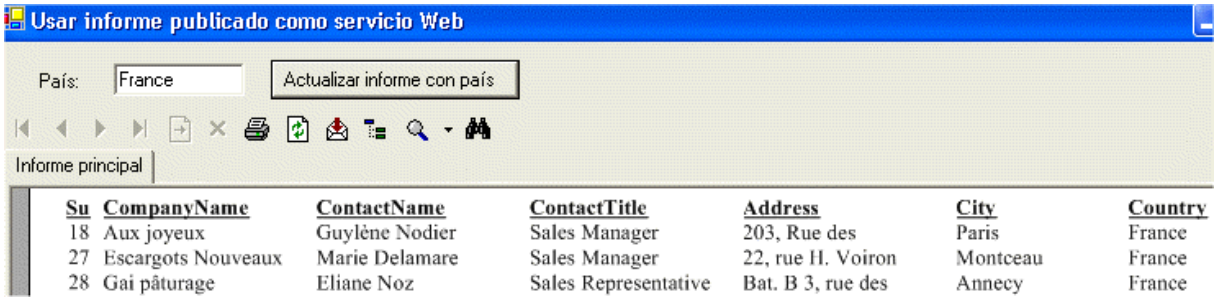


Figura 124. Selección de registros en un informe contenido en un Web Service.

Con este ejemplo, el lector habrá podido comprobar cómo mediante la tecnología basada en servicios Web, el campo de distribución de nuestros informes se amplía de una manera que hasta hace poco era impensable.

Si quiere ver más textos en este formato, visítenos en: http://www.lalibreriadigital.com.
Este libro tiene soporte de formación virtual a través de Internet, con un profesor a su disposición, tutorías, exámenes y un completo plan formativo con otros textos. Si desea inscribirse en alguno de nuestros cursos o más información visite nuestro campus virtual en: http://www.elcampusdigital.com

Si quiere información más precisa de las nuevas técnicas de programación puede suscribirse gratuitamente a nuestra revista ***Algoritmo*** en: http://www.algoritmodigital.com.

Si quiere hacer algún comentario, sugerencia, o tiene cualquier tipo de problema, envíelo a la dirección de correo electrónico lalibreriadigital@alhambra-eidos.com.

© Grupo EIDOS
http://www.alhambra-eidos.com